SC33-0067-0

Customer Information Control System/Virtual Storage (CICS/VS) Version 1, Release 3

Introduction to Program Logic Manual

Program Product

Program Numbers 5740-XX1 (CICS/OS/VS)
5746-XX3 (CICS/DOS/VS)

IBM

First Edition (February 1977)

This edition applies to Version 1, Release 3 (Version 1.3) of the program product Customer Information Control System/Virtual Storage (CICS/VS), program numbers 5746-XX3 (for DOS/VS) ~~and, for planning purposes only~~, 5740-XX1 (for OS/VS).

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using this publication, consult the latest *IBM System/370 Bibliography*, GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments to IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire, SO21 2JN, England. Comments become the property of IBM.

Preface

This publication is an introduction to the internal logic of CICS/VS. The book is intended for programming support representatives and system support programmers who maintain CICS/VS.

This publication is organized into two parts as follows:

- Part 1 provides an introduction to and an overview of CICS/VS
- Part 2 provides a survey of CICS/VS by component and function.

This publication assumes the reader has an understanding of CICS/VS as explained in the publication CICS/VS General Information, GC33-0066.

The following publications also contain information needed when installing and using CICS/VS:

- CICS/VS Application Programmer's Reference Manual (Command Level), SC33-0077 -- for information on coding CICS/VS application programs in ANS COBOL and PL/I.
- CICS/VS Application Programmer's Reference Manual (Macro Level), SC33-0079 -- for information on coding CICS/VS application programs in assembler language.
- CICS/VS Messages and Codes, SC33-0081 -- for messages issued by CICS/VS
- CICS/VS System Programmer's Guide (DOS/VS), SC33-0070 -- for detailed information on generating and starting CICS/VS under DOS/VS.
- CICS/VS System Programmer's Guide (OS/VS), SC33-0071 -- for detailed information on generating and starting CICS/VS under OS/VS.
- CICS/VS Operator's Guide, SC33-0080 -- for a description of the functions performed by the master-terminal operator
- CICS/VS System Programmer's Reference Manual, SC33-0069 -- for information on installing a CICS/VS system
- CICS/VS System/Application Design Guide, SC33-0068 -- for information on designing application programs to execute under control of CICS/VS

For a complete description of the internal logic of CICS/VS consult one of:

- CICS/VS Program Logic Manual (OS/VS), LY33-6029-0
- CICS/VS Program Logic Manual (DOS/VS), LY33-6028-0

Contents

Figures

Part 1. Survey of CICS/VS

This part provides an overview of CICS/VS. It introduces the functions of CICS/VS, the tables which control the operation of the system, and the more important control blocks and tables. There is a general discussion of the execution of CICS/VS.

The part contains five chapters:

Chapter 1. Introduction

The IBM Customer Information Control System/Virtual Storage (CICS/VS) is a data base/data communications (DB/DC) system. In order to understand CICS/VS it is necessary to be aware of the way an online DB/DC system, as opposed to a conventional batch system, processes data.

SYSTEM CHARACTERISTICS

BATCH AND DB/DC SYSTEMS

In a batch processing environment (see Figure 1-1.), application programs are usually scheduled individually to manipulate a batch of data records in sequence. The data files are organized to suit the requirements of a particular application; each application is scheduled independently and the data base support provided by the batch-processing system is unique to each application. The structure of the data is developed for the efficiency of the particular batch program and is not likely to be suitable for a whole range of applications.

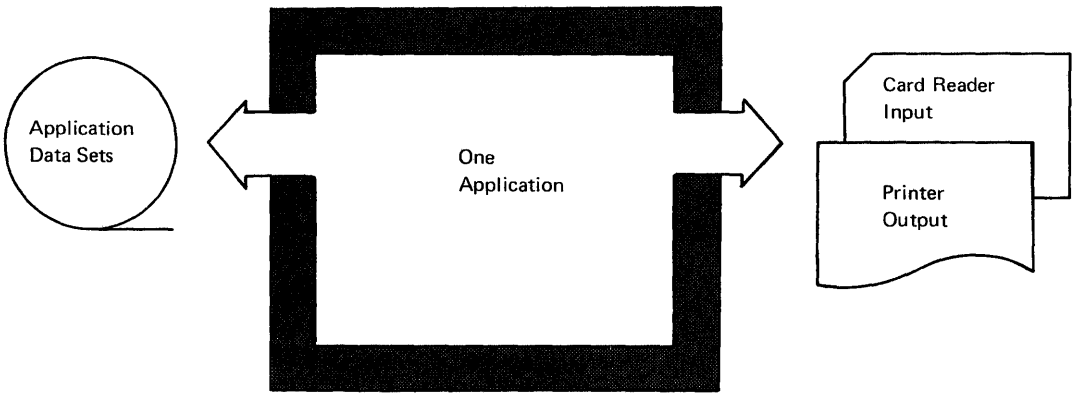

Figure 1-1. Batch Processing

• It is important that a batch job should take as little time as possible so that the installation can process more jobs, but the results of each job are not required within seconds of its entry into the system. In a batch environment, the response time (that is to say, the time between submitting a job and getting the output back) is usually of the order of hours. Again, the entry of batch jobs can usually be planned. The various jobs need not arrive in a completely random and uncoordinated way, and it is possible to organise the installation to take advantage of this and run each job at the most suitable time.

The real-time DB/DC environment (see Figure 1-2.) differs from the batch processing environment primarily in the number and types of concurrent activities that are likely to occur within the system at a given time. Thus, a DB/DC system accepts many requests arriving at

random, and provides a data base organization so that the same data can be used by many applications.

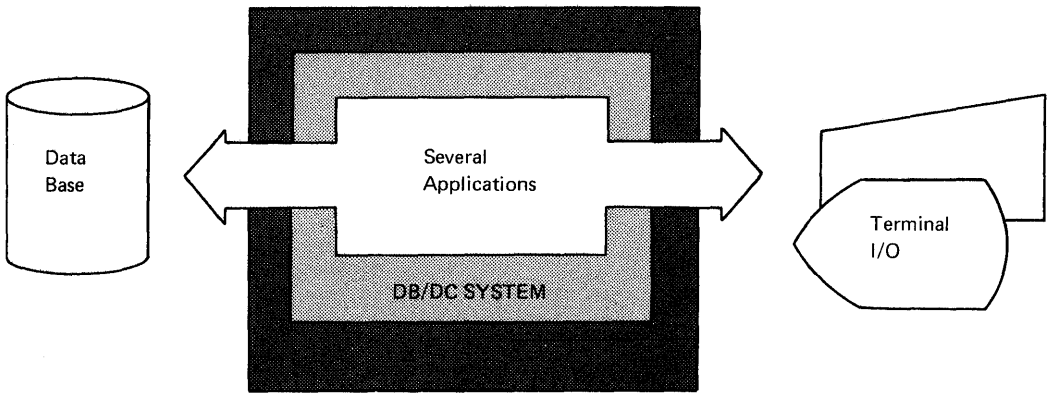

Figure 1-2. On-line Processing

Some of the characteristics of such a system are:

- A DB/DC system is terminal oriented. Application programs executing under the control of a DB/DC system receive input entered from terminals connected to the computer, and send output to the same or different terminals.
- A DB/DC system is capable of servicing two or more application programs concurrently.
- A DB/DC system allows different application programs to access the same data base.

In a real-time system, the aim is not just to keep the computer busy. Since the terminal operators are directly involved in working with the computer, it is important that the response time to a request shall be of the order of seconds, or less. The computer will be subject to fluctuating pressure as the terminal operators submit work. The DB/DC system must therefore have a more complex multiprocessing and priority system in order to allocate work between terminal operators when there is contention between them.

The DB/DC system will run continuously, usually for a day or longer. It may be a single job, a set of jobs running concurrently, or its continuity may depend on information in the data-base. Since it is meaningless to rerun the whole DB/DC system, there must be a restart capability at the level of the unit of processing initiated by a single terminal request. This means that the DB/DC system must be able to keep a record of any changes made.

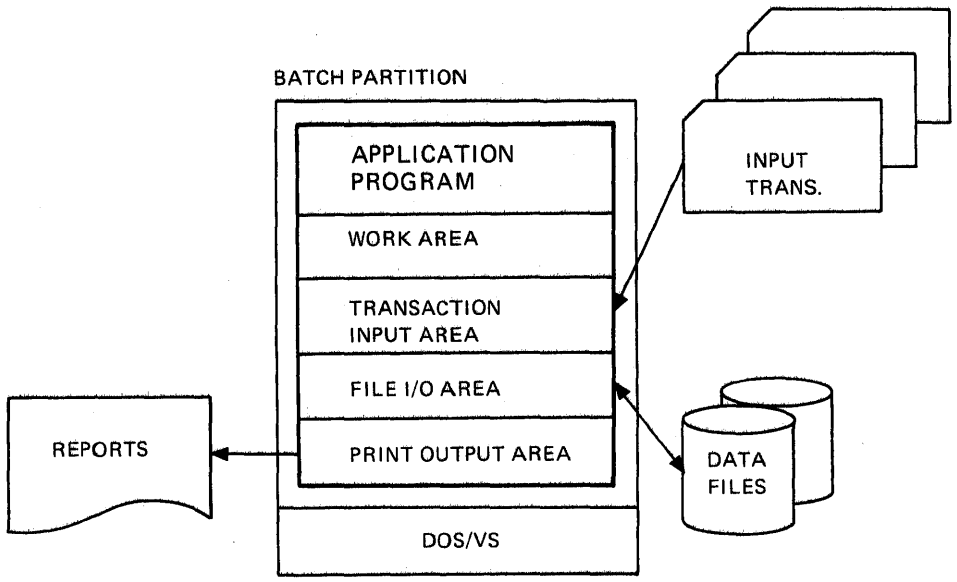

Figure 1-3. Batch Application Program

CICS/VS AS A DB/DC SYSTEM

CICS/VS, then, is a general purpose DB/DC system. It provides support for online systems in much the same way as the operating system and access methods provide support for batch processing systems.

CICS/VS runs as a single job. There may also be other continuously running jobs present, which will usually be part of the operating system. However, CICS/VS is neither a replacement for, nor part of, an operating system. It executes under the control of the Disk Operating System/Virtual Storage (DOS/VS) or the Operating System/Virtual Storage (OS/VS1 or OS/VS2), and uses standard access methods. At the same time as CICS/VS is running other programs can be executed, under the operating system, in other partitions or regions.

CICS/VS acts as an interface between the user's application programs and the operating system. For application programs, CICS/VS provides macro instructions and commands to request services, such as reading and writing files. CICS/VS conveys these requests from the application program to the operating system. In this way, the application programmer is relieved of planning and implementing such input/output requests, of controlling terminals and files or data bases, and of handling abnormal conditions.

CICS/VS may also be regarded as an extension of the operating system. It is executed as one job either in a dedicated mode, with no other partitions operating (except those needed by the operating system to support CICS/VS), or in a multiprogramming mode, with one or more batch partitions. Within its partition, CICS/VS controls the simultaneous processing of input from many terminals by many application programs; CICS/VS therefore has its own task dispatcher, which is logically separate from the control of multitasking in the operating system.

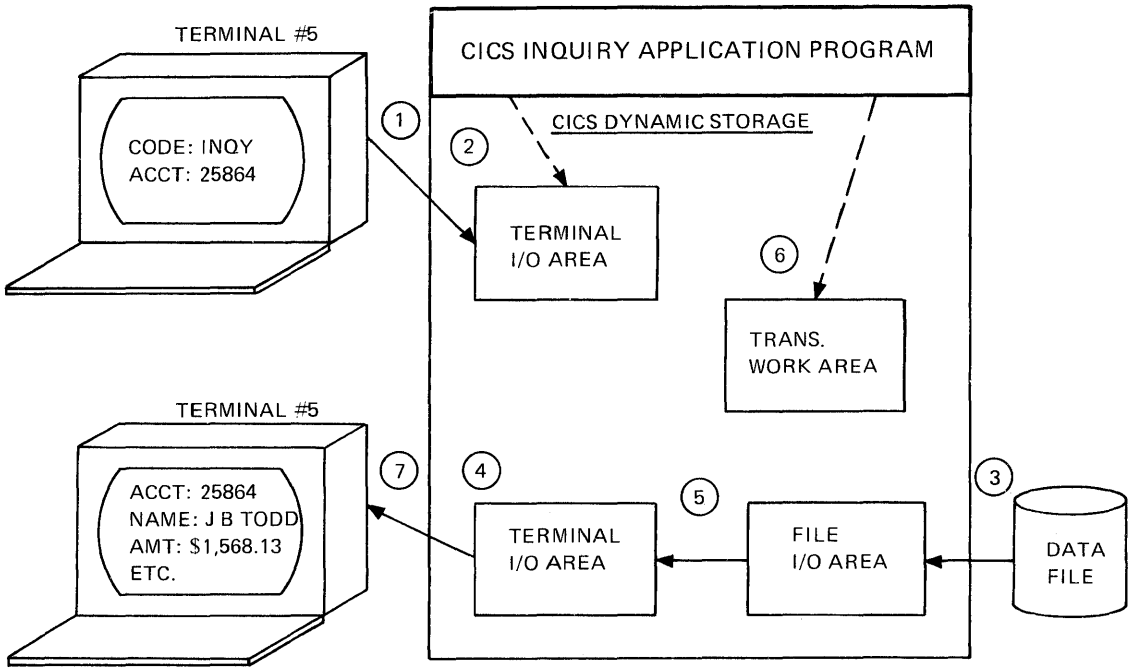

Figure 1-4. CICS/VS Inquiry Application Program

Before a typical application is discussed, some differences between batch and CICS/VS application programs should be reiterated.

- In a batch program, all the I/O and work areas required are defined within the program, or are acquired directly from the operating system. In CICS/VS these areas are outside the application program. I/O and work areas are allocated by CICS/VS when needed, from a dynamic storage area within the CICS/VS partition. (Figures 1-3. and 1-4. highlight this difference.) This allows CICS/VS to conserve main storage, and to process many transactions concurrently.

- A batch program issues I/O instructions directly to the operating system, while a CICS/VS application program issues CICS/VS I/O macro instructions or commands, which are implemented by suitable operating system macro instructions.

INFORMAL INTRODUCTION TO CICS/VS

When CICS/VS is running, it receives a series of inputs from terminals. The unit of processing typically associated with one line or record is a transaction. A transaction will often result in a single reply being sent back to the user's terminal. A transaction may, less usually, involve several input/output operations. When the input that starts a transaction is received from a terminal, a task is created by CICS/VS to control the processing of that transaction until its completion. A task can therefore be defined as the internal CICS/VS representation of a transaction, and is normally associated with the terminal that initiated the transaction. The result of the transaction could be, for example,

an update to a file, or particular information being displayed at the terminal.

CICS/VS handles more than one transaction at a time by overlapping I/O operations and processing. Since CICS/VS resides in a partition with high priority, CICS/VS usually retains control in a multiprogramming environment as long as there are CICS/VS transactions to be serviced. It relinquishes control to the operating system when no further CICS/VS processing needs to be done.

In order to help provide the facilities required of a DB/DC system, CICS/VS is designed as a modular system, made up of packages. The CICS/VS user determines which of these packages he wants, according to the requirements of his particular applications. A CICS/VS system consists of: control modules, system tables, control areas, and user application programs.

The control modules are the programs that implement the CICS/VS macros and commands, and contain the calls to the operating system. The system tables contain a definition of the environment in which CICS/VS is running. The control areas contain the changeable information needed by CICS/VS as it runs. Finally, application programs are written by the user to perform the particular processing of terminal input and of the data base.

The following sections give a general overview of the structure of CICS/VS. The subject is dealt with in more detail in Chapters 2, 3, and 4 of this manual.

CICS/VS CONTROL MODULES

A CICS/VS system contains a number of control modules, some of which are listed below. In descriptions of CICS/VS, its various sections are named in slightly different ways. The standard classification of CICS/VS divides it into components, which are then divided into functions. Many of the control modules are functions within the System Management component. In that context they are generally called, as they are here, Task Management, Storage Management etc.; but in other contexts they may be called Task Control or the Storage Control Program etc.

The first five functions in the list are always present; the remainder are optional and are chosen by the user during CICS/VS system generation.

Task Management -- the dispatcher of the CICS/VS system. Controls the operation of all the tasks active at any one time.

Storage Management -- handles all storage in the CICS/VS partition.

Program Management -- controls loading and releasing, and invocation of CICS/VS application programs.

Terminal Management -- controls all terminal activity.

Time Management -- controls all time services; for example, the suspension of a task for a certain period of time, or the initiation of a task at a particular time.

File Management -- controls the I/O operations needed to support the data base.

Transient Data Management -- provides a queueing facility for data sent to and from user-defined destinations.

Temporary Storage Management -- provides a symbolic scratchpad facility so that an application program can store data, temporarily, in virtual storage or on a direct access device.

Dump Management -- provides a dump of any CICS/VS task (and, on option, of CICS/VS tables).

Journal Management -- provides facilities for creation, management, and retrieval, during real-time CICS/VS execution, of special purpose sequential data sets called journals.

Trace Management -- provides a means of tracing the processing path of an application program.

Basic Mapping Support -- facilitates information display on a wide variety of terminals and provides device independence, terminal paging, and message routing capabilities.

In addition to the principal service functions CICS/VS provides many other online and offline modules, for example:

On line -- System initialization and termination, error recovery, master terminal support, etc.

Off line -- Utilities, formatting programs, system generation, etc.

CICS/VS uses rather than duplicates operating system services. For example, Terminal Management uses telecommunication access methods (BTAM, VTAM, and TCAM) for terminal I/O. File Management uses standard file access methods, such as Indexed Sequential Access Method (ISAM).

CICS/VS TABLES

Associated with some of the management programs are tables, which are generated during CICS/VS Environment Definition, and allow the user to describe or define a particular DB/DC environment to CICS/VS. By using tables in this way CICS/VS can be adapted to work in a different environment merely by supplying different tables.

A full list of the CICS/VS tables, together with a summary of their contents, appears in Chapter 3. Some of the more important tables are:

Terminal Control Table -- used by Terminal Management. It contains descriptions of terminals and features, and operating information.

Program Control Table -- used by Task Management. It defines which transaction codes may be entered by terminal operators, and for each code, the related application program which starts the processing of the transaction.

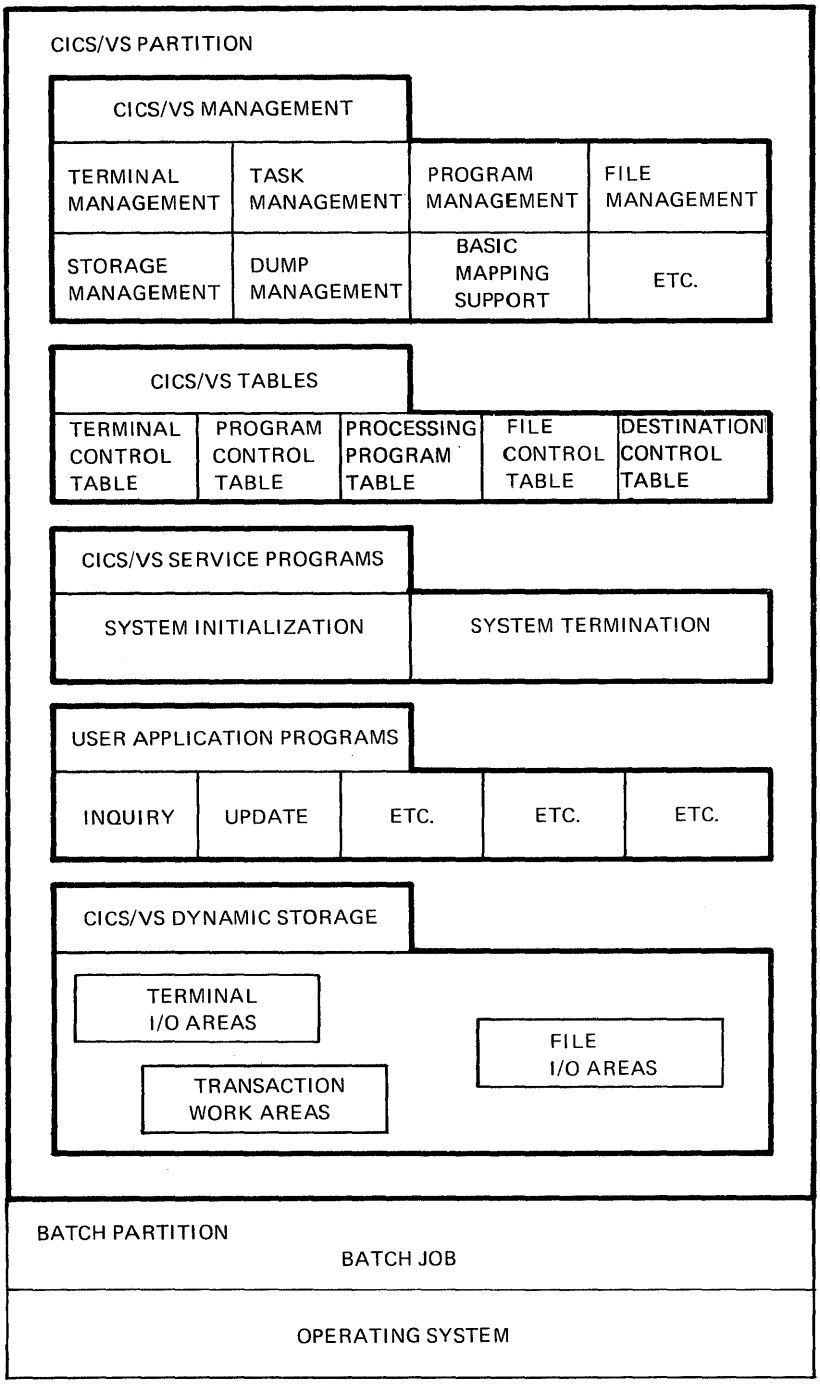

Figure 1-5. CICS/VS in Storage

Processing Program Table -- used by Program Management. It contains information on each application program.

File Control Table -- contains the characteristics of the files accessed by File Management.

Destination Control Table -- used by Transient Data Management. It describes each of the Transient Data destinations used in the system.

Journal Control Table -- Contains information on each of the journal files used in the system.

The File Control Table and Destination Control Table are optional.

CICS/VS CONTROL AREAS

In addition to the CICS/VS system tables, which are used to define the environment to CICS/VS, there are a number of control areas used to hold dynamic data during the execution of CICS/VS. A summary of these areas is given in Chapter 4; the most important are:

- The Common System Area (CSA) is the major CICS/VS control block. It contains pointers to the CICS/VS management modules, control information, and pointers to CICS/VS system tables.
- A Task Control Area (TCA) is created for each task that currently exists. It contains the pointers to the storage associated with the task, pointers to task related fields in other CICS/VS control blocks, current application requests, and save areas.
- A Dispatch Control Area (DCA) is created as a logical extension to each TCA. The DCA is placed in a chain and is used to control task dispatching. It contains information such as the priority and current status of the associated task.
- Transaction Work Areas (TWAs) may, optionally, be present as physical extensions of TCAs.
- Terminal Input Output Areas (TIOAs) are used as buffers for data transmitted to and from terminals.

CICS/VS USER APPLICATION PROGRAMS

The other components of an installation using CICS/VS are the application programs written by the user to provide the on-line processing required by the installation under the control of CICS/VS.

Programs to be run under CICS/VS may be coded in Assembler language, American National Standard (ANS) COBOL, or PL/I. In an assembler language program, the interface to CICS/VS is through macro-assembler instructions, which are expanded by the macro pass of the assembler in exactly the same way as operating system macros. In a COBOL or PL/I program, the interface is a set of commands which are processed by the Command language translator. (There is also a preprocessor which enables low level macro instructions to be used in COBCL and PL/I.) Some of the basic characteristics of CICS/VS application programs are summarized below.

- CICS/VS macro instructions or commands (rather than programming-language statements such as READ, GET, PUT, and WRITE) are included to specify the system and I/O functions required in application

programs. Although the application programmer is not precluded from direct communication with the operating system, CICS/VS will work with greater efficency if it is allowed to perform all supervisory and data management services for the applications.

- Application programs must be coded so that they are serially reusable between CICS/VS macro instructions or commands. A serially reuseable portion of an application program is executed by only one transaction at a time, and must initialize or restore any instructions or data that it alters within itself during execution.

  Programs written with this property needed by CICS/VS are called quasi-reentrant, since the programs need not meet System/370 specifications for true reentrance. Quasi-reentrance allows a single copy of a user-written application program to be used to process several transactions concurrently, thereby reducing the number of copies of a program that must be in main storage. A genuinely reentrant program will always be quasi-reentrant.

- Input/output areas, temporary storage areas, and work areas are not included in an application program. Where these areas are needed they are defined outside application programs, by means of the CICS/VS system tables.

- Files are not defined within application programs.

Figure 1-5. illustrates the general layout of storage when CICS/VS is running.

In order to illustrate how the CICS/VS functions, tables, and areas, and the application programs are related, the following description, under the heading "Example of a Typical Application", follows the processing of a simple transaction step by step.

EXAMPLE OF A TYPICAL APPLICATION

This example considers an inquiry application coded using the macro level interface. Let us suppose a CICS/VS inquiry has been initiated by a terminal operator to answer a customer question. To start an inquiry, the operator keys in a transaction code such as "INQY", which identifies the type of inquiry and the application program which processes it. An account number is also entered and used as a key to identify a particular record. The inquiry application program reads the record, moves the necessary fields to an output area, and displays the information at the terminal.

The following notes, referring to Figure 1-4., provide a simplified view of this transaction; the numbers refer to the diagram, not to the order of events:

1. Terminal Management reads input from a terminal into a Terminal Input/Output Area, within dynamic storage. CICS/VS automatically passes control to the application program identified by the transaction code.

2. The application program refers to an account number in the Terminal I/O Area, and, using it as a key, issues a File Management macro instruction requesting that the record be read.

3. Storage Management allocates a File I/O Area, and File Management issues the read, using the operating system.

4. The application program issues a Storage Management macro instruction to acquire a larger Terminal I/O Area in which to build the message to be displayed.

5. The application program moves the required fields into the new Terminal I/O Area.

6. A Transaction Work Area is made available for working storage, if needed.

7. The application program requests, through a Basic Mapping Support macro instruction, that Terminal Management should write the information to the terminal, concluding the transaction. CICS/VS releases all storage acquired for this transaction for use by others.

So the functions are divided between the application program and CICS/VS as follows:

Application program functions:

- requests that a file be read
- requests that an output Terminal I/O Area be acquired
- building the data to be sent to the terminal
- requests that the message be written to the terminal

CICS/VS functions:

- All I/O, including reading the transaction identifier
- Responses to requests for the storage areas needed to process the transaction

The following descriptions show the CICS/VS functions as they would be used in this typical inquiry application written in assembler language. Each function is first described in general terms and then shown in a diagram. This diagram, which is built up as the transaction is processed, shows at which stage of processing the function is required. Following the diagram, a more technical description of the function is given.

Terminal Management

Assume that the terminal operator has entered the transaction code and an account number. Terminal Management reads this input message into a Terminal Input/Output Area (TIOA) in dynamic storage; see Figure 1-6. Note that Terminal Management performs this read operation, not the application program.

Terminal Management controls all terminal operations through teleprocessing access methods. In this case the access method used is BTAM. Chapters 5 and 6 contain discussions of Terminal Management using VTAM. With BTAM, Terminal Management's primary functions are polling and addressing. Polling is checking all remote terminals periodically to determine whether any have input to transmit, and inviting them to send input to be processed by the application program. Addressing is having the computer check to see if a terminal is ready to receive output. Together with Basic Mapping Support, Terminal Management

provides an application program with the ability to communicate with a terminal. Terminal Management also handles I/O errors, and keeps track of which task is associated with which terminal.

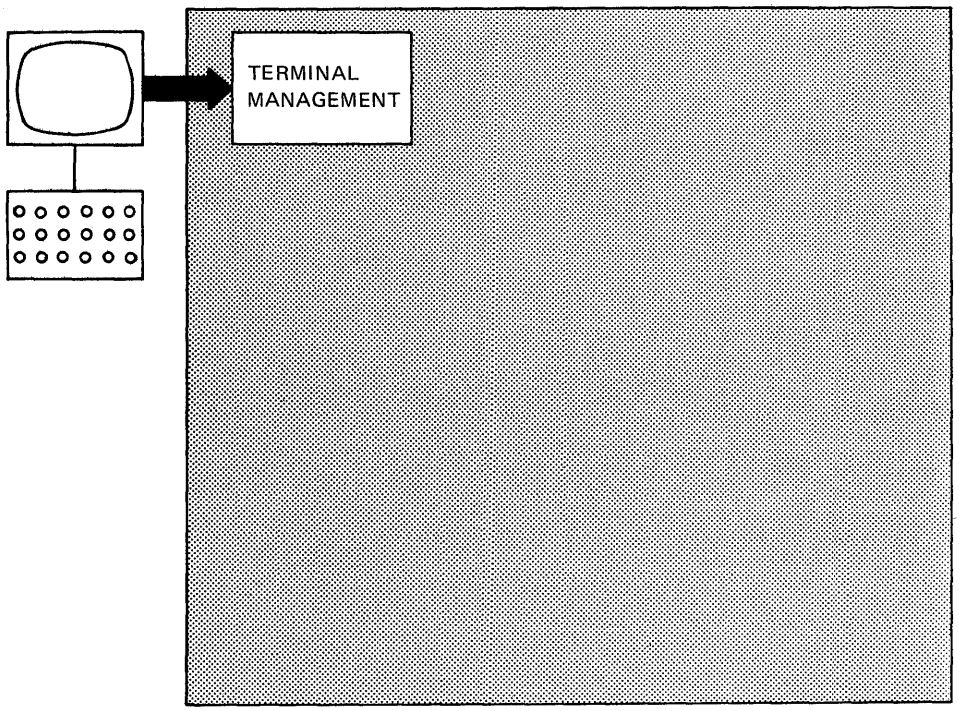

Figure 1-6. Terminal Management

Terminal Management uses the Terminal Control Table (TCT) to help in controlling terminal operations. The TCT specifies the communication line characteristics, the types of terminals, special features, terminal priorities, the polling sequence (the order in which the terminals should be polled) and operational data, such as indications that terminals are temporarily out of service and excluded from polling.

CICS/VS provides several error routines. When unrecoverable I/O errors occur, Terminal Management uses a Terminal or Node Abnormal Condition Program to analyze the condition. Statistics are maintained, and an error message is generated. A user-written Terminal or Node Error Program should be incorporated to tailor the generalized Terminal or Node Abnormal Condition actions to the installation's specific needs.

Task Management

Terminal Management passes control to Task Management, which creates a task for the inquiry transaction (see Figure 1-7.). A terminal can have only one transaction associated with it at a given time, and the terminal is locked until the program writes a response to the terminal.

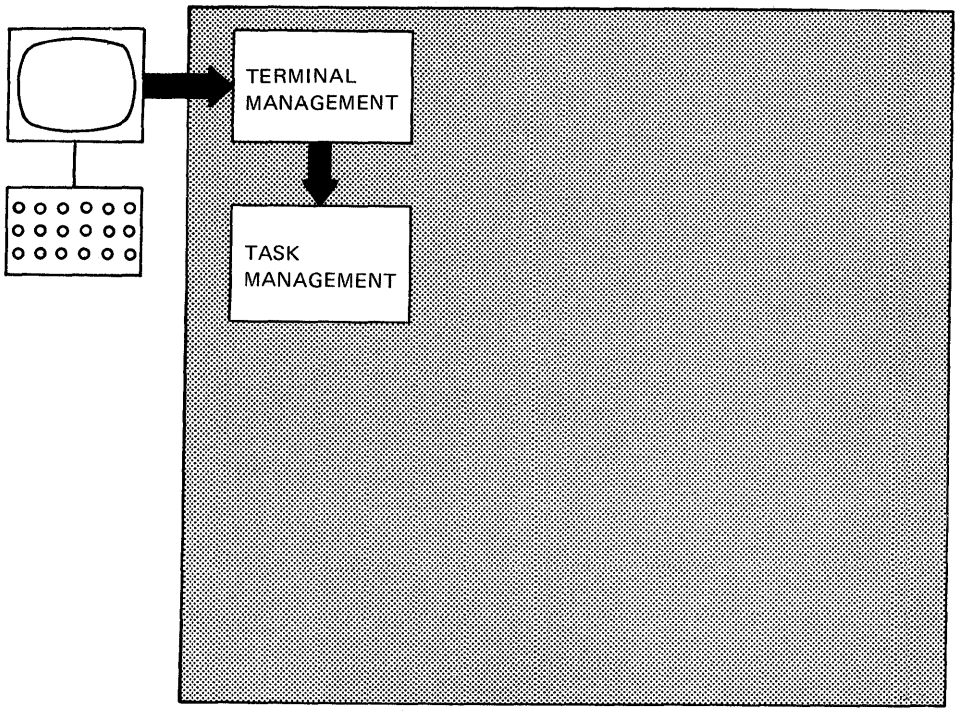

Figure 1-7. Task Management

Task Management keeps track of the status of the many tasks being processed concurrently. Transactions are not usually processed through to completion in a single, uninterrupted operation. A transaction may be processed until a file I/O macro instruction is executed, for instance, whereupon another task receives control. Therefore, there may be many incomplete tasks which Task Management must supervise simultaneously.

CICS/VS has a priority scheme to allocate control of the CPU to the various tasks that execute concurrently. The process of selecting tasks and giving them control is called task dispatching. A user may assign numerical values to transaction codes, the most important having the highest value. Terminal operators and terminals may also be given numerical priorities. Task Management adds these three figures together and, when there is more than one task ready to be dispatched, selects the one with the highest total priority.

Task Management validates transactions by checking the Program Control Table, (PCT), which lists all valid transaction codes and their associated programs, so that control may be transferred to the correct program. If an operator entered an invalid transaction code, Task Management would not find it in the PCT and an error message would automatically be sent to the terminal.

To control each task, Task Management acquires a Task Control Area (TCA) through Storage Management (described later in this chapter). If desired, this area may be extended to include a Transaction Work Area (TWA), which may be used by an application program during the life of a transaction. The TCA and TWA are released when the task terminates.

Program Management

Task Management passes control to Program Management, which keeps track of the locations of the application programs (see Figure 1-8.).

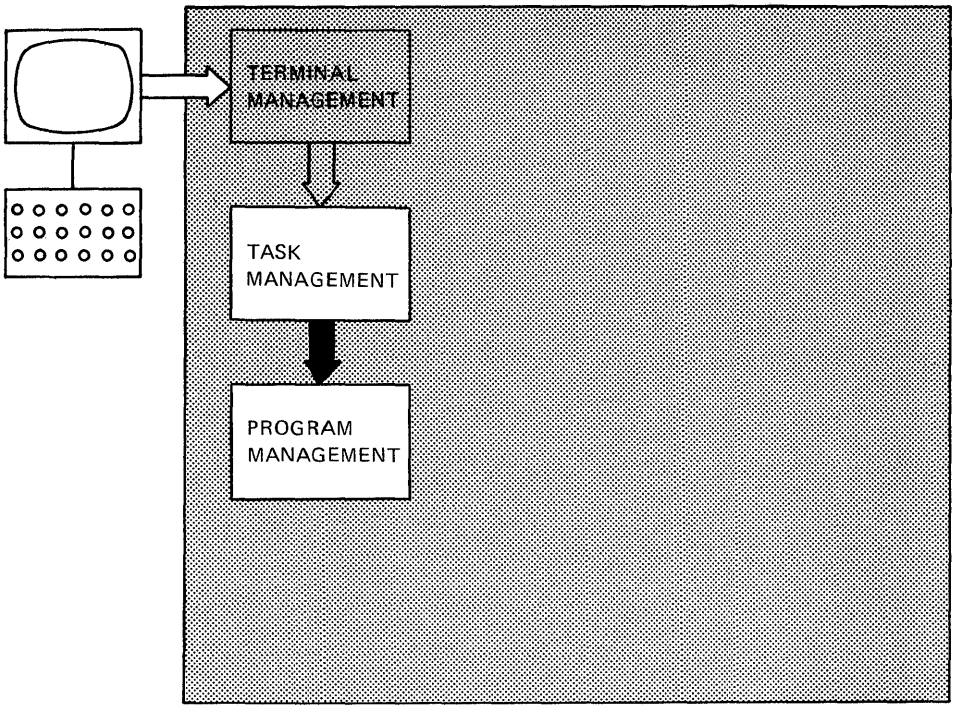

Figure 1-8. Program Management

Program Management controls application programs that are stored in the real-time relocatable library. Programs are loaded into virtual storage when they are needed, unless marked in the Processing Program Table (PPT) as normally resident.

The PPT is used by Program Management to determine a program's location in virtual storage during CICS/VS operation. Programs are relocatable, and may be in different main storage locations from execution to execution. The PPT contains the program size, source language, and other program information.

Under certain conditions, such as unrecoverable I/O errors, the application program may wish to end the task. However, if a program check occurs, this type of abnormal end is treated differently. If a program terminates abnormally in a batch system, the operating system may purge the job in that partition and schedule another. Clearly CICS/VS should not be purged just because of one application program's exception condition. Therefore, CICS/VS intercepts program checks and only terminates the tasks in which they occur.

User Application Program

Program Management passes control to the application program, in this example, the program that handles the transaction code INQY (see Figure 1-9.).

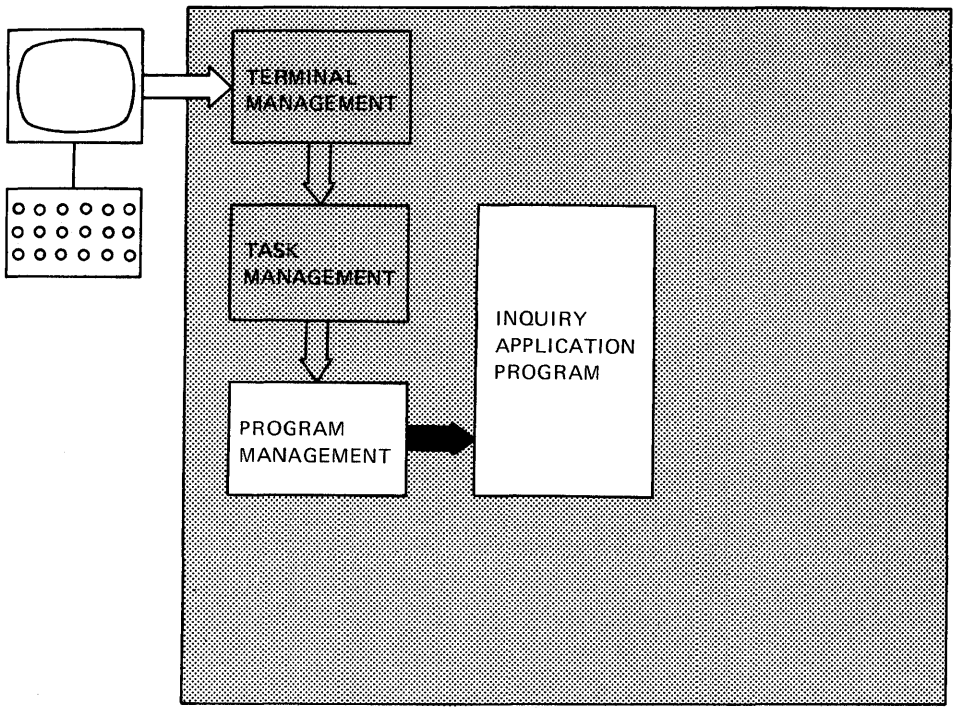

Figure 1-9. User Application Program

In summary, before passing control to the application program, CICS/VS has read the input (INQY and account number) into a Terminal I/O Area, validated the transaction code, and initiated a task. The application program may now process this input and issue macro instructions to request the services needed to handle the transaction.

Basic Mapping Support - Input

Basic Mapping Support (BMS) is a CICS/VS feature that allows the user to define layouts of terminal pages or screens. It is described in more detail in Chapter 11 of this manual. In the context of this transaction description, the next step is as follows:

A BMS macro instruction is issued by the application program to format and move the account number from the Terminal I/O Area to a map area. Control passes to the BMS program to perform this service and returns to the application program (see Figure 1-10.).

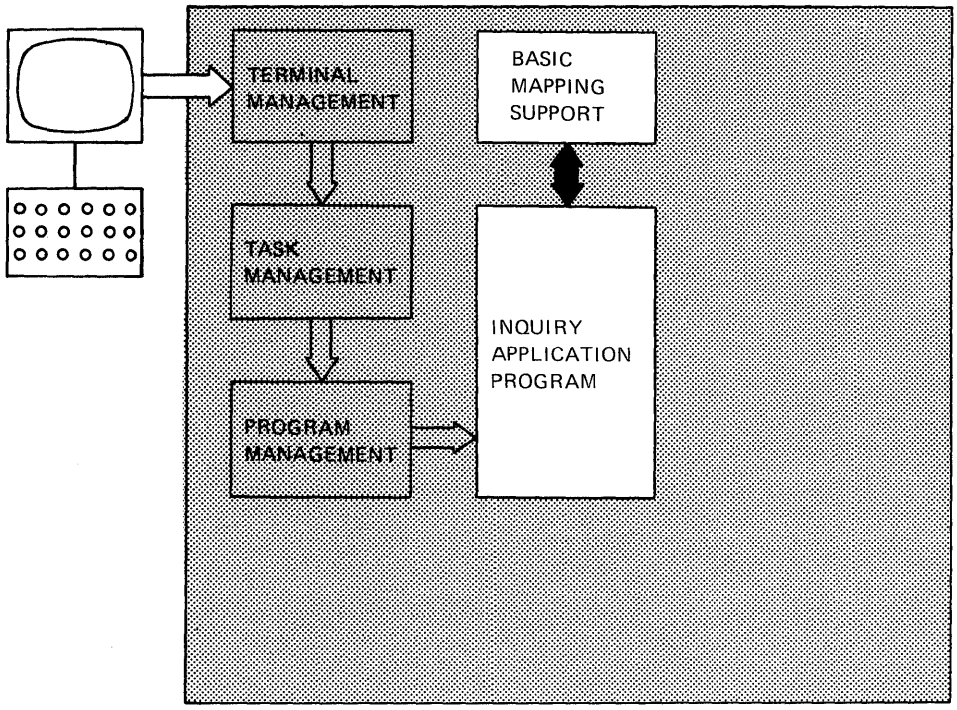

Figure 1-10. Basic Mapping Support - Input

BMS provides two main services to the application programmer, device independence and format independence. Device independence allows an application programmer to communicate with a terminal without having to understand its hardware control characters. Format independence simplifies the positioning of data on the terminal and allows rearrangement of data fields without application program modifications.

BMS uses map tables as requested by the application program to control the formatting (or mapping) of terminal I/O data. A map table is defined for each page or screen layout used in the application program. The map table contains such information as the length and position on the terminal of each data field, visual display field attribute characters, and constant data for headings and keywords.

File Management

The application program issues a File Management macro instruction to retrieve a record from a file or data base. File Management reads the record into a file area (acquired automatically by Storage Management), and returns control to the program (see Figure 1-11.).

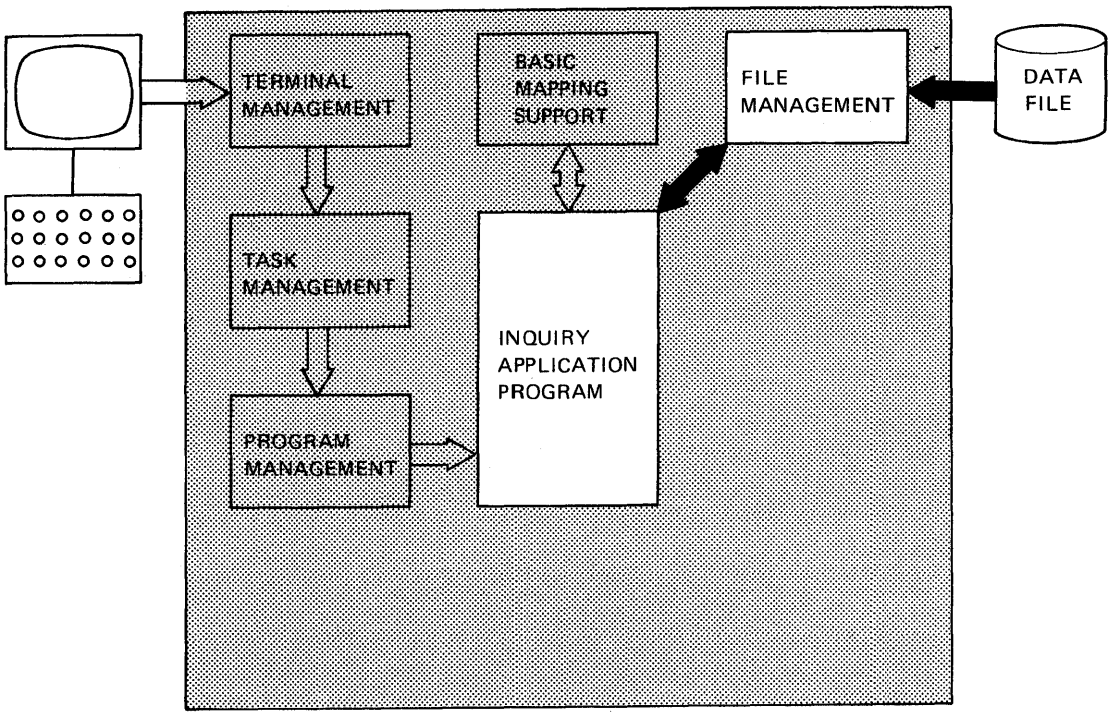

Figure 1-11. File Management

File Management supports read, update, add, delete, and browse functions, if they are supported by the access method being used, and provides a file protection function called exclusive control. This function is invoked by File Management if several tasks request the same record for updating. Exclusive control places all tasks, with the exception of the first, into a wait queue, so that only one task at a time updates that record and returns it to the file before another task may access the record.

The File Control Table (FCT) contains, for each file, user-supplied file characteristics including the access method, record format and length, and block size. The FCT also specifies what operations can be performed on each file. A file may be online and yet effectively protected against modifications by specifying that it is read only, when CICS/VS will prevent any program from updating such a file. New files may be added, old files deleted, and characteristics such as blocking factors modified, without necessarily having to change the application programs.

Transient Data Management

When a file is changed by updating a record or adding a new one, it may be necessary to hold data associated with the change. For example, a copy of the original record may be kept. Such data should be recorded by using Transient Data Management. Inquiries do not change records and would only need to be a source of information if some sort of overall

data, such as statistics, were being collected. Control returns to the application program after the invocation of Transient Data Management.

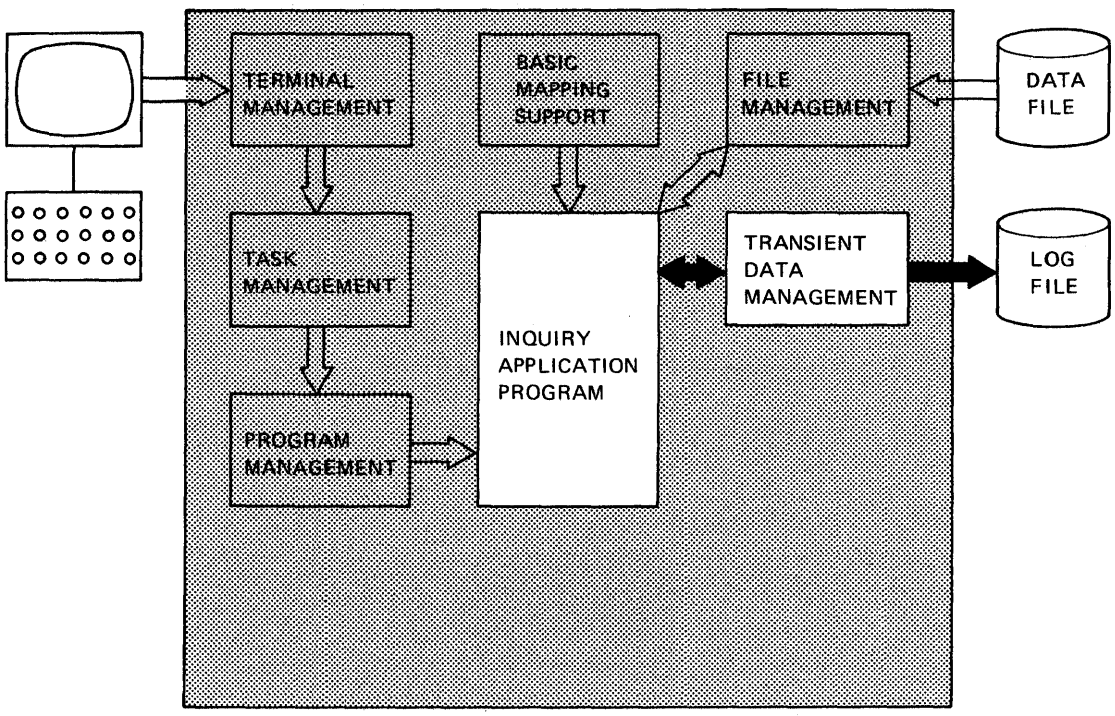

Figure 1-12. Transient Data Management

Transient Data Management is a queuing facility which stores records in the order received on a sequential file or tape file. These may be records for later batch processing, audit records, statistics, or error messages. Sequential input files may also be read by issuing Transient Data macro instructions.

The Destination Control Table (DCT) contains information used by Transient Data Management to direct data to the correct file. The DCT includes the file name, and record and file descriptions.

Trace Management

Trace Management is a CICS/VS debugging aid which may be used to trace the processing path of an application program. It is invoked through CICS/VS trace macro instructions, and after execution returns control to the application program (see Figure 1-13.).

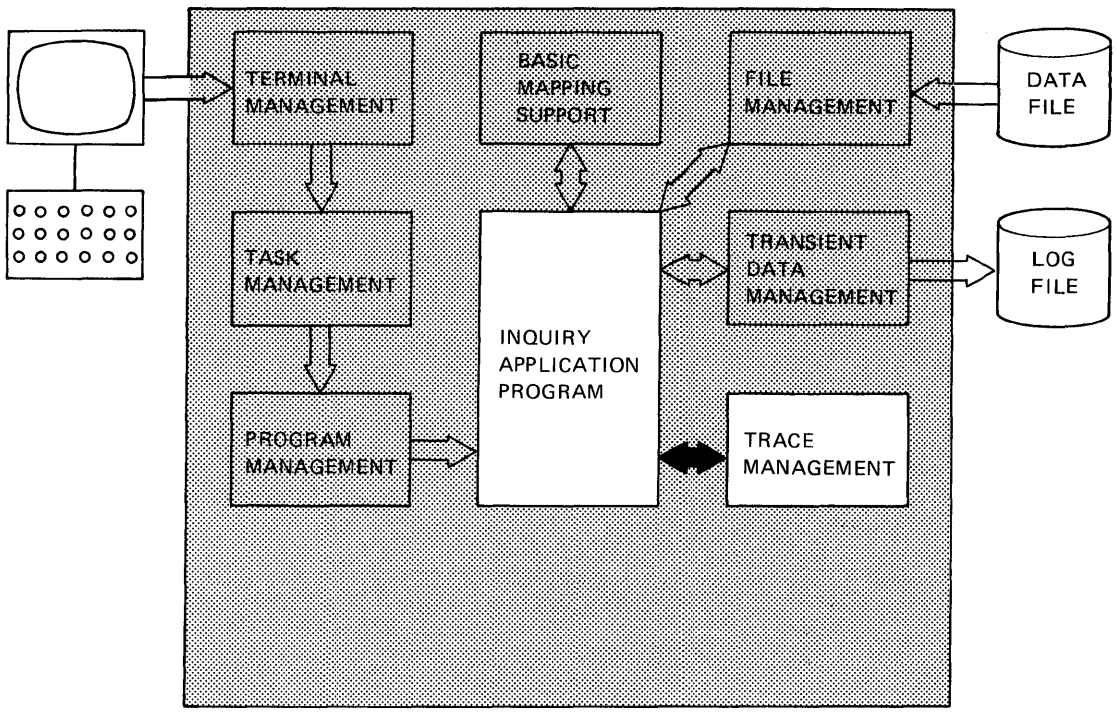

Figure 1-13. Trace Management

When debugging, it may occasionally be difficult to determine which portions of an application program are being executed. By inserting CICS/VS trace macro instructions at appropriate points in a program, the path may be determined. Whenever one of these macro instructions is encountered, CICS/VS makes an entry in the Trace Table. CICS/VS also records in the Trace Table which CICS/VS components have been used by the application program.

The Trace Table resides in virtual storage, and a copy is provided whenever an application program requests a storage dump or the CICS/VS partition is terminated abnormally.

Dump Management

If an unusual condition occurs during processing, the application program may issue a CICS/VS dump macro instruction to write all transaction-related storage areas to a dump file. After the dump has been taken, control returns to the application program (see Figure 1-14.).

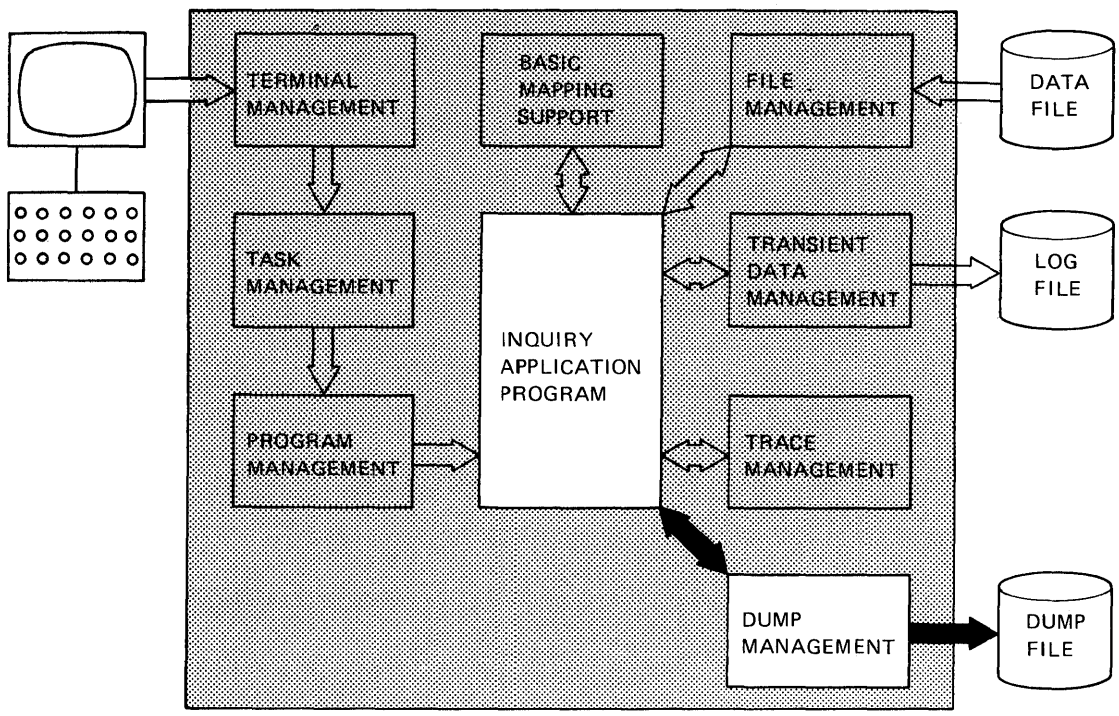

Figure 1-14. Dump Management

Dump Management macro instructions may be inserted at strategic points in a program to facilitate debugging. Dump Management dumps storage areas to a sequential file (DASD or tape) for subsequent printing. The dump macro instructions should be removed from the application program when it has been debugged.

CICS/VS Dump macro instructions may also be used to provide printed records of certain conditions, such as unrecoverable I/O errors. Used in this way, the macro instruction is regarded not as a debugging aid, but as a permanent part of the program.

The Dump Utility Program supplied with CICS/VS is a batch program that formats and prints the dump file.

Temporary Storage Management

The application program may need to store information for later retrieval by another task. Temporary Storage Management allows the program to store such data in virtual storage, or on auxiliary storage on a direct access device, before the next instruction is executed (see Figure 1-15.).

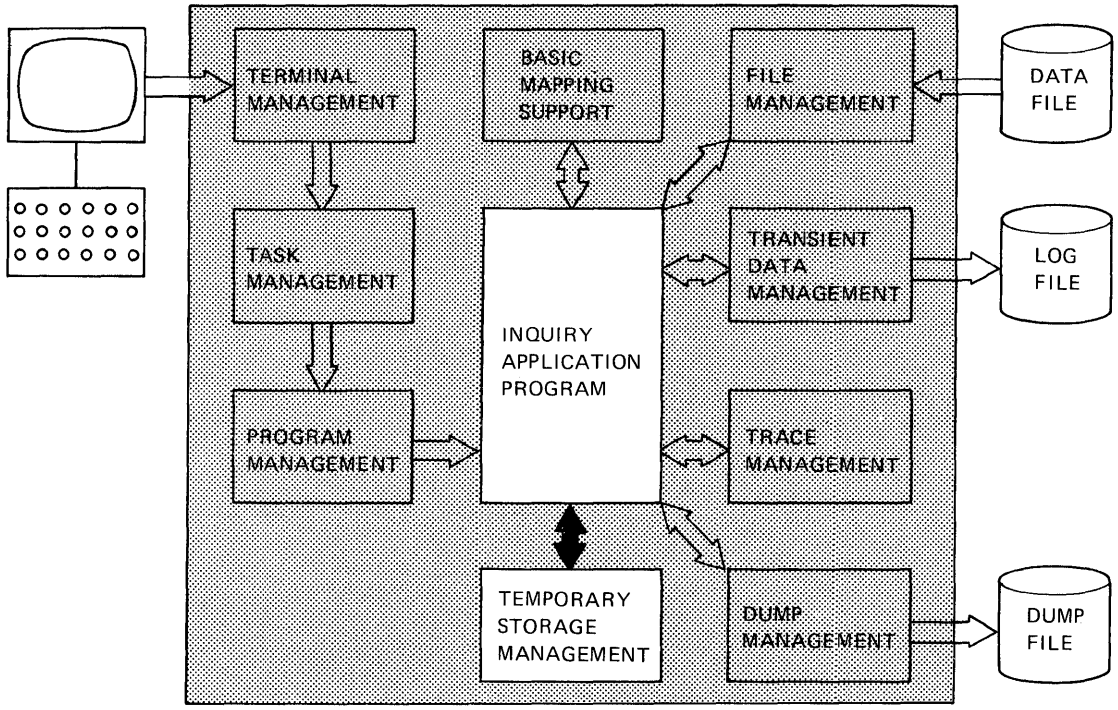

Figure 1-15. Temporary Storage Management

An application program may issue macro instructions requesting that information be stored for subsequent retrieval, assigning a name to each record. If the same name is used to store several records, such records are queued amd may be retrieved later in the same sequence in which they were stored, or randomly by entry number. All temporary storage records, whether single or queued, are retained by CICS/VS until purged by an application program.

Storage Management

The application program must now extract the necessary fields from the record in the file area and set up a map area to be written to the terminal. An output area in which to build the map data is requested by issuing a Storage Management macro instruction. Control returns to the program when the area is secured (see Figure 1-16.).

Storage Management controls virtual storage within the CICS/VS partition or region. Through macro instructions, it allocates storage to other CICS/VS control programs or to application programs. Obtaining a Task Control Area for Task Management for example, is an illustration of allocating storage for other CICS/VS functions.

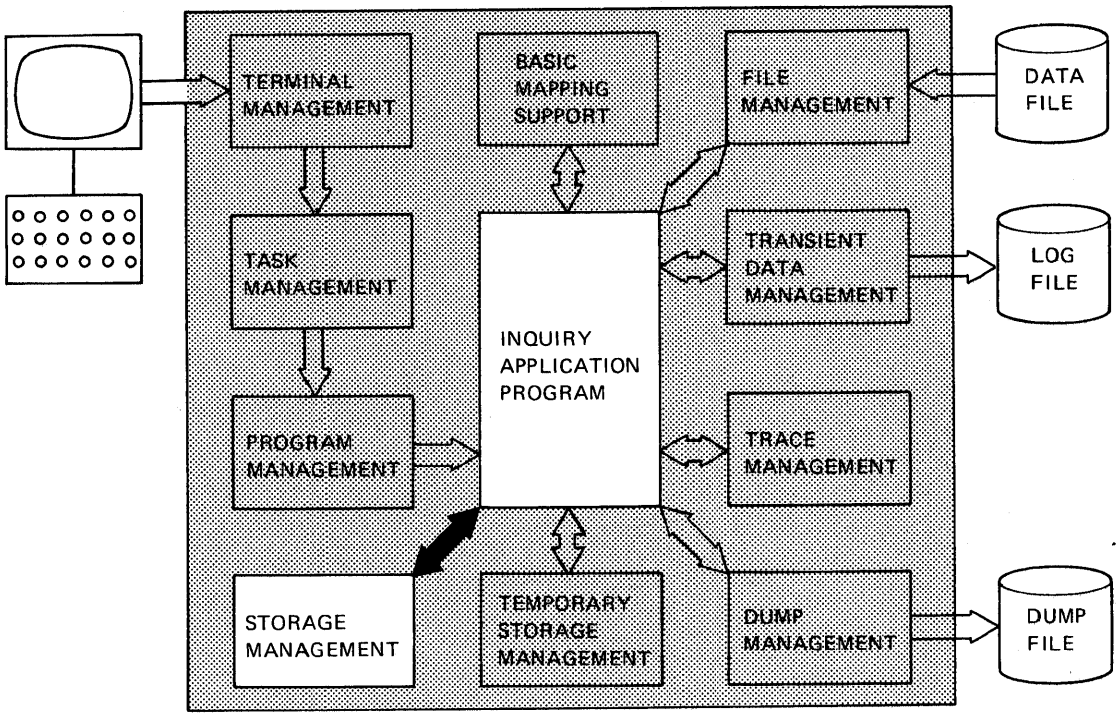

Figure 1-16. Storage Management

To service as many terminals as possible, CICS/VS conserves main storage through dynamic storage management. Instead of preallocating static I/O areas to the tasks, Storage Management assigns those areas from dynamic storage when they are needed.

Because of dynamic storage, allocation locations may differ between transactions. Since CICS/VS programs need to access fields within these areas, Storage Management provides addressability by passing the area locations to the application programs. Storage Management also queues requests for storage, if space is temporarily unavailable. A task is placed in a wait state by Task Management until space becomes available. All storage areas acquired for a task are chained together off one of the CICS/VS control blocks for the task, allowing Storage Management to release the storage on task termination. The application program may also return storage to CICS/VS when it is no longer needed. This permits the storage to be used by other transactions in the CICS/VS partition.

Basic Mapping Support - Output

A BMS macro instruction formats the record fields for transmission to the terminal. BMS moves the data from the map area to a Terminal I/O Area, and it is then written to the terminal by Terminal Management. The terminal operator may view the record for as long as desired, and then enter a new transaction code and initiate a new task (see Figure 1-17.).

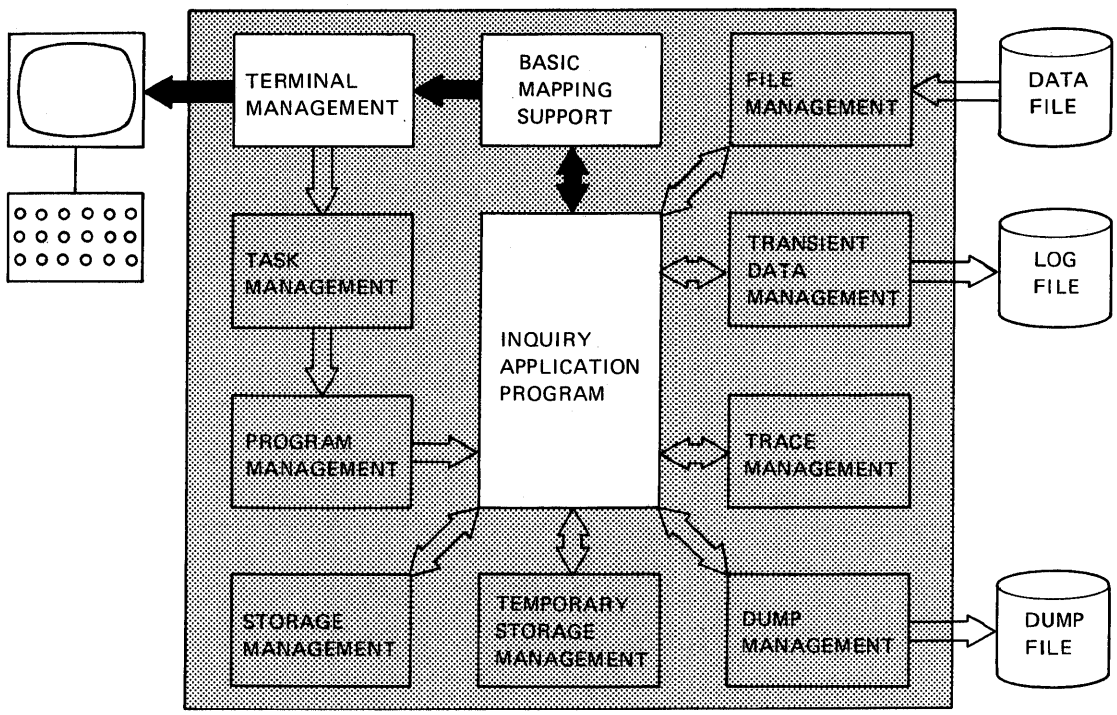

Figure 1-17. Basic Mapping Support - Output

Ending the Transaction

The transaction is terminated by issuing a macro instruction to Program Management. This uses Storage Management to release all the storage allocated to the transaction, and so makes it available for use by other transactions. Upon completion, control returns to Task Management which deletes this transaction from its transaction list. Task Management then invokes Terminal Management to poll the terminal for a new transaction code (see Figure 1-18.).

PLAN OF THE MANUAL

The remainder of this book is intended to provide a description of the operation of CICS/VS in rather more detail. The first part deals with general topics, while the second part covers CICS/VS function by function.

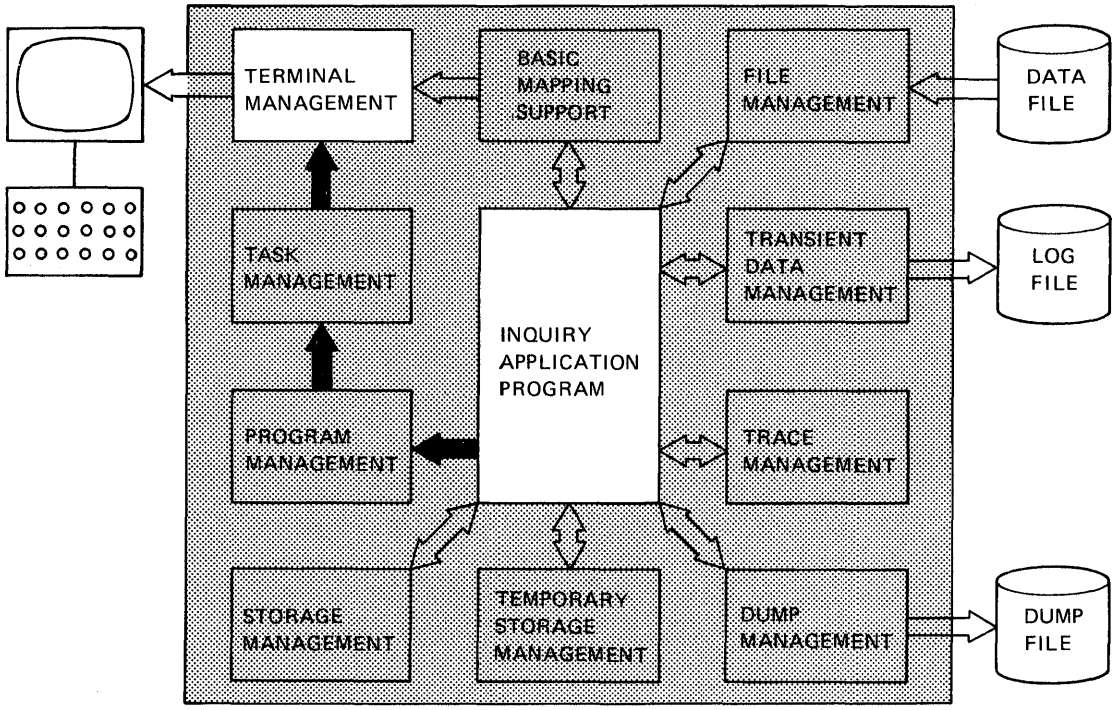

Figure 1-18. Ending the Transaction

Chapter 2 lists the components and functions, stating the purpose of each function. Chapter 3 describes system preparation, giving details of the various system tables. Chapters 4 and 5 cover the execution time operation of CICS/VS. Chapter 4 deals with systems with a single CPU, and Chapter 5 deals with CICS/VS in a network with one or more controllers.

If you are trying to get an overall picture of CICS/VS, you should continue to read the book sequentially. If you are interested in a specific area, you should now have enough background knowledge to continue with the chapter in which you are interested.

Chapter 2. CICS/VS Structure

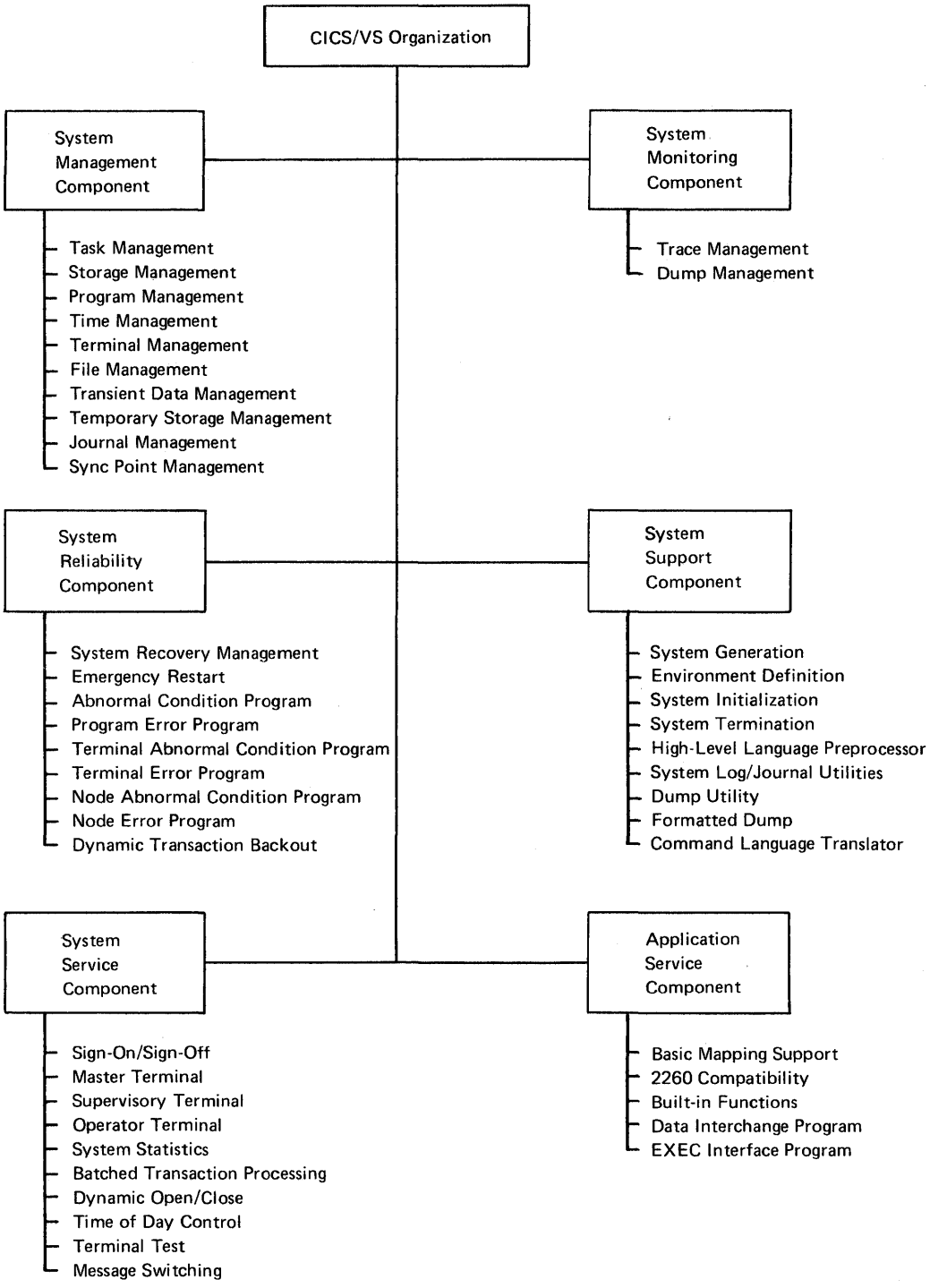

Figure 2-1. CICS/VS Organization - Components & Functions

The previous chapter introduced the basic requirements of a data base/data communications system, and described, through a general overview of the modular structure of CICS/VS and an example of a typical application, how CICS/VS meets these requirements. This chapter describes the organization of CICS/VS in more detail, concentrating on the purpose of each CICS/VS function.

The classification followed in this chapter (which appears throughout the CICS/VS manuals) is also used in the chapters of Part 2 of this manual, where the internal logic of CICS/VS is summarised.

CICS/VS is logically structured into six major components, each of which contains a set of functions, as shown in Figure 2-1. These functions, in turn, provide services to the CICS/VS user. Most services are requested directly by the application programs through CICS/VS macro instructions, but some, which help to create a useful DB/DC environment, are performed automatically by CICS/VS.

The great majority of CICS/VS functions are either part of the CICS/VS nucleus, that is to say they are an integral part of the system and are (virtually) loaded at system initialization time, or they are sytem application programs which are loaded when needed in the same way as the users programs are loaded.

SYSTEM MANAGEMENT

System Management contains most of the functions that are central to the running of CICS/VS. All the functions are resident in the nucleus. The first four of the following functions are supervisory; the remainder are concerned with data management.

TASK MANAGEMENT

Task management controls the allocation of CPU time between contending CICS/VS tasks. It is the analogue of the central dispatcher in an operating system, providing an interface to the multitasking facilities of the host operating system. It also provides some services to the application programs: attaching tasks, synchronizing tasks, and queueing for resources.

STORAGE MANAGEMENT

Storage Management controls the virtual storage allocated to CICS/VS and to the user-written application programs. Services provided by storage management include the acquisition, initialization and disposition of storage.

PROGRAM MANAGEMENT

Program Management controls programs within CICS/VS. The services provided include multiprogramming, the loading, linking, and deletion of programs, and the transfer of control between them.

TIME MANAGEMENT

Together with Task Management, Time Management (sometimes called Interval Control) provides various optional task functions (system stall detection, runaway task control, task synchronization, etc.) based on specified intervals of time, or the time of day.

TERMINAL MANAGEMENT

Terminal management provides the communications between terminals and user-written application programs. The Basic Telecommunications Access Method (BTAM), Virtual Telecommunication Access Method/Network Control Program (VTAM/NCP), and Telecommunication Access Method (TCAM) are used for most terminal data management and line control services. Terminal Management supports automatic task initiation to process transactions which use a terminal but are not directly initiated by the terminal operator. It also provides a simulation of terminals by sequential devices in order to help test new applications.

FILE MANAGEMENT

File management provides a data base facility using keyed access through the Virtual Storage Access Method (VSAM), Indexed Sequential Access Method (ISAM), and the Direct Access Method for OS/VS (BDAM) or DOS/VS (DAM). File Management supports updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data on the data base. CICS/OS/VS provides multithread access to the Data Language/I (DL/I) facilities of the IBM Information Management System (IMS/VS). CICS/DOS/VS provides multithread access to Data Language/I DOS/VS. These CICS/VS DL/I interfaces allow CICS/VS application programs to access DL/I data bases.

TRANSIENT DATA MANAGEMENT

Transient data management provides an optional queuing facility for managing data being transmitted between user-defined destinations(I/O devices or CICS/VS tasks). This function facilitates data collection.

TEMPORARY STORAGE MANAGEMENT

Temporary Storage Management provides an optional general purpose scratchpad facility. It is intended for video display paging, broadcasting, data collection, and retention of control information.

JOURNAL MANAGEMENT

Journal management provides facilities for creation, management, and retrieval of special purpose data sets, called journals, during real-time CICS execution. Journals are intended for recording, in

chronological order, any data the user may need later in order to reconstruct data or events. For example, journals could be created to act as audit trails; to record data-base updates, additions and deletions for backup; or to track transaction activitity in the system.

SYNC POINT MANAGEMENT

Sync Point Management allows the user to specify a point in the application program which is the end of a logical unit of work. Any processing performed before such a sync point will not be reversible if there is an error after the sync point.

A sync point is also taken automatically at the end of each task.

SYSTEM SERVICES

The System Services component contains a number of ancillary application programs that provide system service functions. Although several of these are designed to be optional, the functions and services are extremely valuable to the running of a DB/DC system in an installation.

SIGN-ON/SIGN-OFF

This function provides terminal operator identification to give more security.

MASTER TERMINAL

The Master Terminal program provides dynamic user control of the system. A master terminal operator can change the status and values of parameters used by CICS/VS and thereby alter the operation of the system. He may temporarily disable entries in several CICS/VS tables and terminate any CICS/VS task currently in the system.

SUPERVISOR TERMINAL

The supervisor terminal function performs a terminal-oriented subset of the services available to the master terminal. These services are limited to the terminals under a given supervisor's control.

OPERATOR TERMINAL

This function allows a terminal operator to control the service and processing status of the terminal.

SYSTEM STATISTICS

This function provides the capability for CICS/VS to log system statistics.

ASYNCHRONOUS TRANSACTION PROCESSING

Asynchronous Transaction Processing allows the user to read batched input from an appropriate device, storing it in a queue. The data can then be taken from the queue, processed and the results written to another appropriate device. Asynchronous transaction processing is performed concurrently with other terminal activity. Although designed specifically for batched terminals (like the 2780, 3780, and 2770), this feature can be used with certain interactive terminals (for instance the 2741).

DYNAMIC OPEN/CLOSE

The Dynamic Open/Close function allows the user to open and close data sets during the real-time execution of CICS/VS.

TIME OF DAY CONTROL

Time of Day Control helps CICS/VS to run continuously for more than 24 hours. CICS/VS adjusts the expiration times that it maintains in response to changes in the time of day maintained by the operating system, and then resets its own date and time of day to that of the operating system.

TERMINAL TEST

Terminal test is primarily designed for Field Engineers to help them instal new terminals during the real-time execution of CICS/VS. Upon request all printable characters can be sent, or a message can be 'echoed'.

MESSAGE SWITCHING

This function provides the user with a general purpose message switching capability while CICS/VS is running. The facility, which can route messages to one or more destinations, is initiated by the transaction code 'CMSG', or a user-chosen replacement, read from the terminal.

SYSTEM MONITORING

The System Monitoring component consists of two functions that run on-line and provide diagnostics to the user. Both functions are resident in the CICS/VS nucleus.

TRACE MANAGEMENT

Trace Management provides a program debugging facility that records the execution of CICS/VS macro instructions by CICS/VS management and service programs, and by user-written application programs.

DUMP MANAGEMENT

Dump Management provides help in analysing programs and transactions that are being developed or modified. Specified areas of dynamic storage are dumped onto a sequential data set, either tape or disk, for subsequent offline formatting and printing, using the CICS/VS dump utility program. See also the Formatted Dump program.

SYSTEM RELIABILITY

The functions in the System Reliability component handle error conditions and help the user to recover or restart after an error occurs.

SYSTEM RECOVERY MANAGEMENT

System Recovery Management enables CICS/VS to intercept program interrupts and abnormal terminations by the host operating system, in order to prevent the termination of the whole CICS/VS system. If Dump Management is used, a formatted dump is provided. If possible, only the individual task causing the error condition is terminated. System Recovery is resident in the nucleus.

DYNAMIC TRANSACTION BACKOUT

Dynamic Transaction Backout allows the effects of an abnormally terminating transaction to be reversed imediately, while the rest of CICS/VS continues normally.

ABNORMAL CONDITION

This program resolves any abnormal conditions other than those associated with a terminal, or those handled directly by the operating system.

PROGRAM ERROR

Each CICS/VS installation may supply a routine to provide a user action in response to a programming error. CICS/VS provides the option of disabling the transaction code associated with the program in error, thus preventing the recurrence of the error until it can be corrected.

TERMINAL/NODE ABNORMAL CONDITION

These functions intercept any terminal/node abnormal conditions that are not handled by the operating system.

TERMINAL/NODE ERROR

The user may supply routines to provide corrective action in response to terminal or node I/O errors. A sample error program is supplied on the CICS/VS distribution tape or disk.

EMERGENCY RESTART

The Emergency Restart function allows the user the option of restarting CICS/VS following an abnormal termination (machine check, power failure, or abnormal termination by the operating system), and reinitializing CICS/VS selectively in order to meet his requirements.

KEYPOINT PROGRAM

The Keypoint Program collects information from tables and control areas, and writes the information onto the Restart data set, or the System log, for use by System Intialization and Emergency Restart.

SYSTEM SUPPORT

The System Support component consists of several functions that are required to support the real-time CICS/VS system. Most of the support functions are performed off-line.

SYSTEM GENERATION

System Generation enables the user to define and structure CICS/VS to meet the particular requirements of the installation. System generation is controlled through CICS/VS system generation macro instructions.

ENVIRONMENT DEFINITION

Environment Definition enables the user to create control tables and service tables that define the environment in which the generated CICS/VS system is to operate.

SYSTEM INITIALISATION

System Initialization is used to start the CICS/VS job. The facility is resident only long enough to bring CICS/VS into storage and start up its execution.

SYSTEM TERMINATION

The System Termination program allows the user to end the current operation of CICS/VS. The function will gather summary statistics and information necessary for a warm start, and then return control to the operating system.

HIGH-LEVEL LANGUAGE PREPROCESSOR

The off-line HLL preprocessor performs part of the process of preparing a COBOL or PL/I program with embedded CICS/VS macro instructions

COMMAND LANGUAGE TRANSLATOR

The off-line Command language translator prepares a source program, in ANS COBOL or PL/I, with its embedded EXEC CICS commands, for input to the appropriate compiler. It translates the commands to statements in the high level language. There is an interface routine between the commands and CICS/VS which is part of the Application Services component.

DUMP UTILITY

The off-line Dump Utility program formats and prints the output from CICS/VS Dump Management. It operates in batch mode and allows each storage area, program, and table entry to be identified, formatted, and printed separately, with actual and relative addresses.

TRACE UTILITY

The offline Trace Utility program formats and prints the output from Trace Management

SYSTEM JOURNAL FORMATTING UTILITIES

The System Journal Formatting utilities preformat magnetic tapes or disk extents to be used as system logs or journals. They allow the user to place an end-of-file mark on magnetic tapes used as journals, before using them after an abnormal system termination.

FORMATTED DUMP PROGRAM

The Formatted Dump Program may be run when an error occurs. It produces a dump of the CICS/VS partition or region with the various CICS/VS control tables and areas identified.

APPLICATION SERVICES

CICS/VS provides several functions designed to perform services closely associated with user applications. These services rely on CICS/VS System Management functions to achieve their objectives and can be considered as logical extensions to the user-written application programs.

BASIC MAPPING SUPPORT

Basic Mapping Support provides message routing, terminal paging, and device independence services. Message routing allows application programs to send output messages to one or more terminals not in direct control of the transaction. Terminal paging allows the user to prepare a multi-page output message without regard to the physical size of the output terminal; the output can then be retrieved by page number in any order. Device independence allows the user to prepare output without regard to the control characters required for a terminal; CICS/VS automatically inserts the control characters and eliminates trailing blanks from each line. Most of the BMS programs are resident in the CICS/VS nucleus

DATA INTERCHANGE PROGRAM

The Data Interchange program supports the batch controller functions of the IBM 3790 communication system - Stage 4, and also for the IBM 3770 programmable communication system. Support is provided for the transmit, print, message, user, and dump data sets of the IBM 3790.

2260 COMPATIBILITY

This function allows the user (with BTAM) to run currently operational 2260-based transactions from an IBM 3270 Information Display System. Compatibility mode is specified by the user (for a transaction or for a terminal); operation in this mode can be intermixed with IBM 3270 native mode. Two levels of compatibility are provided: a full screen operation

or a format mode. The latter is more efficient; however, not all 2260 operations are supportable within the format mode. The level of support can be selected by transaction. In most cases, the user is not required to make any changes to application programs.

EXEC INTERFACE PROGRAM

The EXEC Interface program analyses the arguments of the an ANS COBOL or PL/I CALL statement, generated by the Command language translator, to determine the requested function and to assign values into the appropriate CICS/VS control blocks. It also relieves the application programmer of storage management and error checking.

BUILT-IN FUNCTIONS

CICS/VS provides the application programmer with some commonly used functions, invoked through the CICS/VS macro-level interface. The built-in function program is resident in the nucleus.

- Table Search

  This provides a convenient means of searching a table for a specific entry, and having some value within that entry returned; if the desired entry is not in the table, the user can elect to have a default value returned.

- Phonetic Conversion

  This provides a method of converting a name into a key based on the sound of the name; the function allows the user to organise and access data sets based on names that might be misspelled, mispronounced, or misunderstood.

- Field Verify

  This function enables the user to verify the contents of a data field as entirely numeric, alphabetic, or packed decimal data.

- Field Edit

  This provides a means of removing alphabetic or special characters from numeric fields and converting the result to EBCDIC or to packed decimal format.

- Bit Checking

  This function allows a COBOL application program to set or test the value of a single bit in storage.

- Input Formatting

  This provides a means of converting free-form input from the terminal operator into a predefined fixed format that can be manipulated more easily; the free-form input may be positional or keyword oriented.

- Weighted Retrieval

  This allows the user to search a group of records on a VSAM key sequence data set, selecting only those records that satisfy specified criteria.

Chapter 3. System Preparation

Before CICS/VS can run, the system must be built up to match the requirements of the user and the machine environment. The two main steps in CICS/VS system preparation are System Generation and Environment Definition, which are broadly described below, and described in detail in the CICS/VS System Programmer's Guides and CICS/VS System Programmer's Reference Manual. System Generation establishes the features of CICS/VS that will be present and Environment Definition involves the creation of tables describing the installation. As well as these main steps, the CICS/VS user must have a suitable version of DOS/VS or OS/VS, and must have defined the various data sets that CICS/VS will require (see 'Data Sets' in Chapter 4 of this manual).

Because CICS/VS is both modular and table oriented, maintenance is simplified. If a change is made in the user's environment which, in turn, requires a change to a CICS/VS management program or table, only the affected program or table needs to be generated again. This is also true of any corrections that must be applied to the system. To make a modification to a particular program, it is only necessary to update the source program, reassemble it, and link-edit the resulting object program.

CICS/VS SYSTEM GENERATION

The CICS/VS user can define a version of CICS/VS that meets his particular needs. The process allows him to select just those functions required by his applications and it is flexible enough to allow a partial regeneration of an existing system. The user can add or delete functions as his applications change.

CICS/VS System Generation comprises two stages. Stage 1 assembles the CICS/VS generation macro instructions and produces the job control and source statements for Stage 2, which is the assembly, link-editing and cataloging of the CICS/VS modules.

The various functions listed in Chapter 2 are provided during CICS/VS system generation. The following functions are required:

- Task Management
- Storage Management
- Program Management
- Time Management
- Terminal management
- Master terminal
- System Termination
- Operator Terminal
- System Statistics
- Time of day Control
- Trace Management
- Abnormal Condition
- Terminal Abnormal Condition

The following functions are optional:

- File management
- Transient data management
- Temporary storage management
- Journal management
- System recovery management
- Sign on/sign off
- Batch transaction processing
- EXEC Interface program
- Recovery/restart support programs
- Dynamic Open/close
- Terminal Test
- Message Switching
- Dump Management
- Basic Mapping Support
- 2260 Compatibilty
- Built-in Functions
- Supervisor terminal

The utility programs needed to support the CICS/VS operation can also be selected during system generation. These include the dump and trace utility programs, the high-level language preprocessors, and the journal format program.

The CICS/VS system generation assembly macro-instruction has the operation code DFHSG. The major keyword parameters of the macro are TYPE and PROGRAM. The sequence of macros is:

```
DFHSG TYPE=INITIAL, ...
DFHSG PROGRAM= ... , ...
DFHSG PROGRAM= ... , ...
 ...
DFHSG TYPE=FINAL
```

The DFHSG TYPE=INITIAL macro is used to specify overall features of the CICS/VS system and to control the system generation assembly itself. The DFHSG PROGRAM=xxx macros select and modify the various CICS/VS functions. A DFHSG TYPE=FINAL macro should be the last macro in the System Generation input.

To build directories for the CICS/VS-DL/I interfaces there are two more System Generation assembly macros: DFHDLPSB and DFHDLDBD.

THE PROGRAM PARAMETER OF DFHSG

The programs that make up CICS/VS are selected by DFHSG macro instructions in which the PROGRAM keyword parameter is set to a code indicating the chosen module or modules. These codes and groups are:

PROGRAM=CSO - Control System Operational Group

- System Initialization
- System Termination
- Abnormal Condition
- Terminal Abnormal Condition
- Dummy Terminal Error Program
- Time Adjustment program
- System Statistics
- Formatted Dump Program
- Dummy Program Error Program
- Message Switching

PROGRAM=CSS - Control System Service Group

- Sign On/Sign Off
- Terminal Test

PROGRAM=CSD - Control System Dummy Group

This consists of a set of dummy programs corresponding to optional management programs. If the management program is not present in the system, then its corresponding dummy program will be included.

PROGRAM=CSU - System Utilities

PROGRAM=KCP - Task Management

PROGRAM=SCP - Storage Management

PROGRAM=PCP - Program Management

PROGRAM=SRP - System Recovery

PROGRAM=ICP - Time Management

PROGRAM=DCP - Dump Management

PROGRAM=TBP - Transaction Backout Program

PROGRAM=TCP - Terminal Management

PROGRAM=RSP - Resend Program

- Node Abnormal Condition Program
- Node Error Program

PROGRAM=FCP - File Management

PROGRAM=TDP - Transient Data Management

PROGRAM=TRP - Trace Management

PROGRAM=TSP - Temporary Storage Management

PROGRAM=JCP - Journalling Management

PROGRAM=CSA - Common System Area

Although the CSA is not a program it is generated in the same way as a program.

PROGRAM=MTP - Master Terminal

PROGRAM=OCP - Dynamic Open/Close

PROGRAM=GAP - Graphic Attention Program

PROGRAM=EXP - Command language translator

PROGRAM=EIP - EXEC interface program

PROGRAM=ATP - Asynchronous Transaction Processing

PROGRAM=KPP - Keypoint Program

PROGRAM=BMS - Basic Mapping Support

PROGRAM=BFP - Built-in Functions

PROGRAM=DIP - Data Interchange Program

SIMPLIFIED SYSTEM PREPARATION

Some assistance is available in order to lessen the work of producing a full CICS/VS system with the standard System Generation. For CICS/DOS/VS the Subset option is provided which enables the user to generate a subset of CICS/DOS/VS more easily. System Generation is simplified because the full range of options is not present. The subset option is described in the CICS/VS Subset User's Guide.

Pregenerated CICS/VS systems are also provided on the distribution tapes. They are built out of pregenerated CICS/VS modules and tables, which are prepared by standard System Generation runs. Pregenerated modules are supplied in the form, or forms, in which they are most frequently used. The pregenerated systems include:

- DOS Subset
- DOS Full Function
- OS Full Function

ENVIRONMENT DEFINITION AND SYSTEM TABLES

CICS/VS is table oriented. This gives the user flexibility in describing terminal, data base, and queuing environments. It also allows him to describe several versions of existing environments, simplifying later conversion to more extensive systems. Because of this table orientation, the user need change only the definition of that part of the environment which has changed.

All tables are specified and constructed using the CICS/VS table generation macro instructions in a macro-assembly. Tables are used to describe system initialization and terminal, data set, and queuing environments, together with user programs, transactions, operators, and so on.

The tables are divided into control and service tables. The control tables fall into two groups, mandatory and optional. The mandatory control tables are:

System Initialization Table	SIT
Terminal Control Table	TCT
Program Control Table	PCT
Processing Program Table	PPT

The optional control tables are:

Temporary Storage Table	TST
File Control Table	FCT
Destination Control Table	DCT
Journal Control Table	JCT
System Recovery Table	SRT

The service tables are:

Nucleus Load Table	NLT
Application Load Table	ALT
Sign-on Table	SNT
Terminal List Table	TLT
Transaction List Table	XLT
Program List Table	PLT

To allow a CICS/VS system that has been generated in a particular way to be used with several environments, alternative system tables are allowed. Each of the system tables, except the Sign-on Table, has, in the CICS/VS system, an eight character name of the form DFHxxxyy. DFH is constant, "xxx" is the table mnemonic given above, and "yy" is a suffix that the user assigns in order to name one version of a table, so that the version may be loaded, on request, during System Initialization. The Sign-on Table is always called DFHSNT.

CONTROL TABLES

The control tables allow the user to identify each element of the real-time environment, the attributes of each element, and the treatment he wishes given to each. Several of the tables are considered essential, while others are optional and, like the CICS/VS management programs, are dynamically selected by the user system initialization.

Program Control Table (PCT)

The Program Control Table is used to define transactions. An entry must be included to identify each transaction code that is to be invoked by any source within the system. Additional attributes applicable to each transaction are also included in the table entry. The following are examples of information within each table entry.

- Transaction priority and security identification
- Transaction work area (TWA) length, used to determine the size of the TWA to be acquired for this transaction
- Initial processing program identification
- Transaction statistics
- Transaction purge status (purgeable or nonpurgeable)
- Transaction read time-out requirement
- Usability status (enabled/disabled)
- Transaction characteristics (inquiry, data collection, conversational,message protection options, test, etc.)
- Associated node error program (NEP)

Processing Program Table (PPT)

The Processing Program Table is used to define application programs. There must be one entry in the table for each application program, signifying that it is valid for execution within CICS/VS and that the program is located in the CICS/VS real-time program library. The PPT also holds attributes of the application program, such as:

- Program source language (Assembler language, ANS COBOL, or PL/I)
- Program residency status (resident or nonresident)
- Usability status (enabled/disabled)
- Force pageout requirement

Terminal Control Table (TCT)

The Terminal Control Table is used to define the user's terminals. It is composed of Terminal Control Table Terminal Entries (TCTTEs). In general, there must be an entry in the table to describe each one of the following:

- Communication line groups
- Communication lines
- Terminals
- TCAM process queues

For the VTAM-supported part of the network, however, the TCT contains only terminal parameters.

Parameters applicable to each communication line group include the following:

- Device address (for sequential devices)
- Error recovery/recording options
- Special device-dependent features (optional)
- Binary synchronous device options

Parameters applicable to each communication line include the following:

- Access method
- Terminal types associated with the line
- Length of input message area
- Device-dependent options and features

Parameters applicable to each terminal include:

- Identifying symbol

- Terminal priority
- Terminal type
- Terminal address
- Initial terminal status

System Recovery Table (SRT)

The System Recovery Table contains information needed by the system recovery program to handle abnormal conditions resulting from errors signalled by the host operating system. The basic SRT provided by CICS/VS can be altered by the user as needed.

System Initialization Table (SIT)

The SIT supplies System Initialization with the information needed to set up the user's environment. During initialization, the user is given the opportunity to change some parameters.

The information contained in the SIT may be grouped into three categories:

- Information used to initialize and control system functions (for example, storage cushion size and system partition/region exit time interval).
- Module suffixes used to load user-specified versions of the CICS/VS control modules and tables (for example, DFHFCPxx and DFHFCTxx).
- Special information used to control the initialization process.

The user can generate several SITs and then select the appropriate one during System Initialization by specifying a table suffix.

File Control Table (FCT)

The File Control Table is an optional table used to describe the structure of the user's data base. There must be one entry in the table for each data set the user wishes to access during the execution of CICS/VS. In addition to providing a symbolic identification for each data set, the user can specify the following attributes:

- Data set organization (ISAM, VSAM (ICIP or normal), BDAM or DAM, or DL/I)
- Accessing options
- Data set characteristics (for example, block size)
- Indirect accessing or indexing control information
- Record segment definitions
- Usability status (enabled/disabled)

- Automatic journaling and logging
- VSAM shared resource characteristics
- VSAM ICIP information (MVS only)

Destination Control Table (DCT)

The Destination Control Table is used to describe to CICS/VS the characteristics of data to be processed by Transient Data Management. There must be an entry in the table for each extrapartition or intrapartition symbolic destination.

If system statistics and error conditions are to be logged by CICS/VS, the appropriate destinations must have entries in the DCT. If the DCT is omitted, system statistics and error conditions are not logged.

Extrapartition data is either generated externally to the CICS/VS environment and processed within CICS/VS, or generated within CICS/VS and processed externally (for example, batch mode).

Intrapartition data is generated and processed within the CICS/VS environment.

In addition to providing a symbolic identification for each destination, DCT entries include the following information:

- Destination characteristics (for example, block size)
- Automatic transaction initiation information
- Indirect destination specifications
- Usability status (enabled/disabled)
- Recovery attributes

Journal Control Table (JCT)

The Journal Control Table is used to describe the user's journal data sets to CICS/VS. There must be one entry in the table for each data set the user accesses during execution of CICS/VS; and in each entry there must be a symbolic identification for each data set and the length of the buffer area to be used.

Temporary Storage Table (TST)

The Temporary Storage Table is used to describe data sets for which temporary storage recovery is to be provided. The TST is a list of generic mnemonics used to identify data sets for which CICS/VS is to provide recovery in the event of abnormal termination of CICS/VS (and subsequent emergency restart) or abnormal termination of a transaction (and subsequent transaction backout)

SERVICE TABLES

The service tables, which are optional, provide the user with increased control over the operation of CICS/VS. The service tables reside in a program library and are loaded into dynamic storage when they are needed.

Sign-On Table (SNT)

The Sign-on Table contains authorization information and is accessed by the CICS/VS Sign-on program when a terminal operator initiates the sign-on procedure.

During sign-on, the name of the terminal operator is entered at the terminal and is used to locate the appropriate operator entry in the table. This operator entry contains data used to verify the operator name and to establish a priority and a security key for the transactions that the operator subsequently enters.

The priority value assigned to the operator is used to develop the task priority for processing a transaction. The operator's security key is used in a security check of all transactions subsequently entered. The security key in the appropriate Terminal Control Table (TCT) entry for the operator is matched with the transaction security contained in the transaction's Program Control Table (PCT) entry.

If the operator security key matches the security value in the PCT entry, the transaction is accepted and the appropriate application program is initiated. Otherwise, a security violation has occurred and the transaction is terminated. A security key of 1 is the default option in the creation of the SNT and the PCT, and is present in the TCT until altered by a sign-on procedure. The security key default option allows transactions with a transaction security key of 1 to be entered into the system by the operator without the sign-on procedure.

The SNT is also used to specify an operator class that augments the message routing facility.

Terminal List Tables (TLT)

The Terminal List Tables enable the user to put the terminals into logical groupings. A logical grouping of terminals could be, for instance, all the terminals that are under the control of one supervisor. Each supervisor can have a TLT, and all terminals that serve a similar function can represent a logical grouping under a supervisor. The user can create a grouping of terminal identifications to facilitate the broadcasting of general messages to terminals.

The system service programs of CICS/VS use TLTs to change the status of a terminal. A unique TLT must be created for each supervisor who is to have the ability to alter the status of the terminals under his control. Any operation to change the status of an entire group of terminals requires a TLT containing the identification of each terminal in that group. A unique identification is assigned to each table by the user.

Program List Tables (PLT)

The Program List Tables contain program identifications that perform various specified functions within CICS/VS. Each program identified must also appear in the Processing Program Table (PPT), making the PLT a subset of the PPT. The PLTs provide the following:

- List of programs to be executed during the post-initialization phase of system initialization.
- List of programs to be executed during the first quiesce stage of system termination.
- List of programs to be executed during the second quiesce stage of system termination.

By providing suffixes for PLTs, the user can have many versions of these tables.

Transaction List Tables (XLT)

The Transaction List Tables contain identifications of CICS/VS transactions. Each transaction in a XLT must also be defined in the Program Control Table (PCT). Thus, a XLT represents a subset of the PCT.

The primary purpose of the XLT is to identify transactions that remain eligible for execution during the 'first quiesce' stage of system termination. By providing suffixes for XLTs, the user can have many versions of these tables.

Application Load Table (ALT)

The Application Load Table allows the user to control the order in which application programs are loaded during system initialization. This facility helps the user to make efficient use of virtual storage.

The ALT is optional; if it is not used, application programs will be loaded in an order based on parameters in the Processing Program Table (PPT). If the ALT is used, the programs specified therein will be loaded first.

Nucleus Load Table (NLT)

The Nucleus Load Table is used by CICS/VS to control the order in which the modules of the CICS/VS nucleus are loaded. The table allows the user to change the order established by the CICS/VS system initialization program. This facility helps the user to make the most efficient use of virtual storage.

The NLT is optional; if it is not used, the nucleus modules will be loaded in the order established by a default NLT which is part of System Initialization.

OTHER REQUIREMENTS

The configuration required to use CICS/VS as a data base/data communication interface is largely determined by the scope of the environment to be supported and the nature of the user's applications.

The configuration must include sufficient I/O devices to support the requirements for system output, system residence, and system data sets. Appropriate line adapters and telecommunication control units are needed. Sufficient direct access storage must be provided to satisfy user information storage requirements.

ACCESS METHODS

CICS/DOS/VS requires a telecommunication access method (BTAM, VTAM, or EXTM), the Direct Access Method (DAM), and the Sequential Access Method (SAM). CICS/OS/VS requires a telecommunication access method (BTAM, VTAM, or TCAM), the Basic Direct Access Method (BDAM), and the Queued Sequential Access Method (QSAM). If Temporary Storage Management is used, the Virtual Storage Access Method (VSAM) is also required. Depending on the user's data base configuration, either VSAM or the Indexed Sequential Access Method (ISAM) may be required.

Data sets created through any of these access methods can be operated on by application programs under the control of CICS/VS and by offline batch-processing programs.

Chapter 4. CICS/VS Real Time Execution Environment

The purpose of this chapter is to describe the way CICS/VS supports an application program. Figure 4-1. shows how the functions mentioned in Chapter 1 form an intermediate stage between the application program and the operating system; and the diagram could be extended by adding all those other CICS/VS functions which are invoked through the application interface. The diagram could also be simplified to Figure 4-2 and it is this simplification that forms the basis of this chapter.

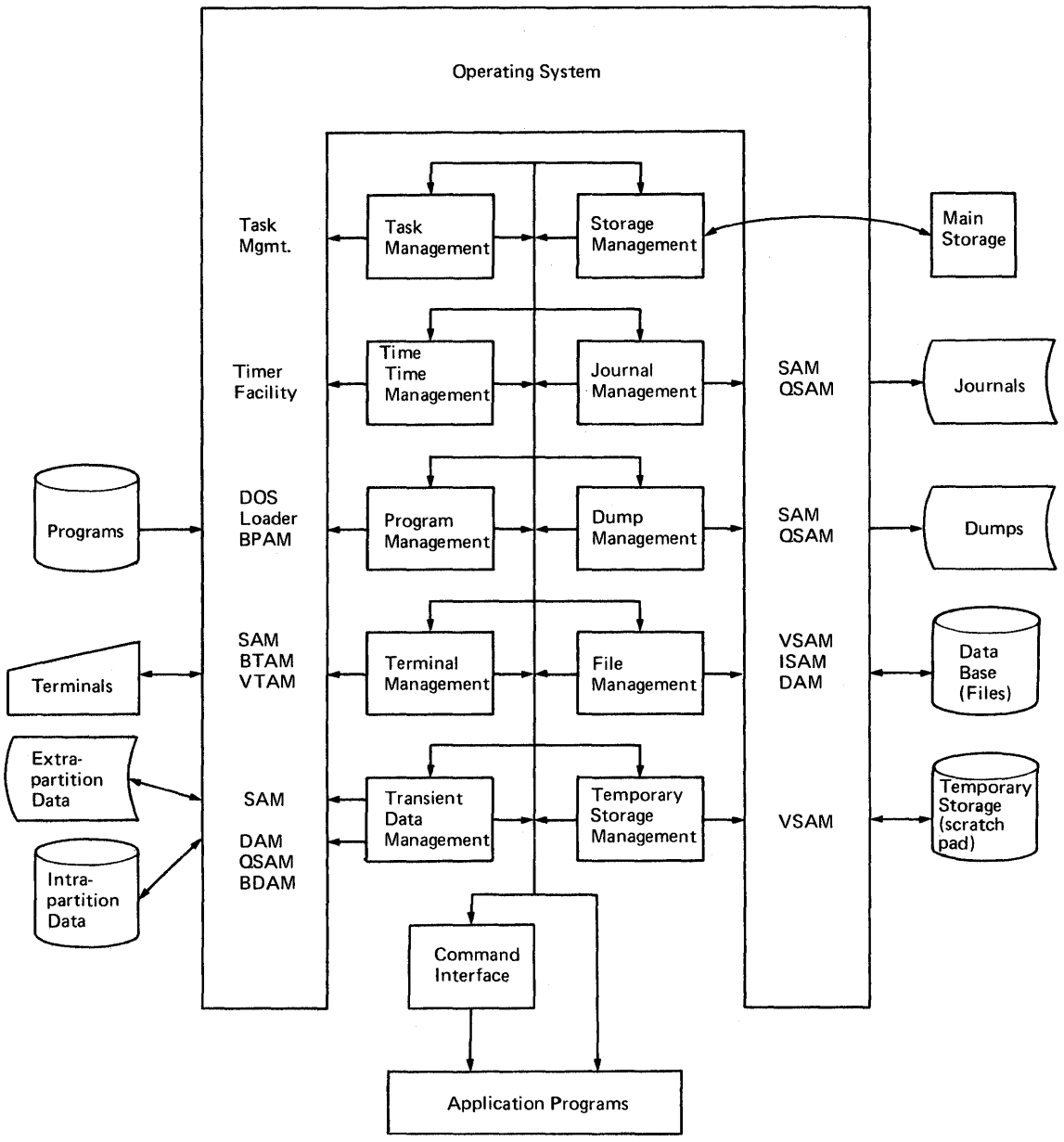

Figure 4-1. CICS/VS and The Operating System

THE APPLICATION INTERFACE

The application program is written in Assembler language, ANS COBOL or PL/I. There are two forms of the application interface:

- The macro-level interface.

  This is directly available to the assembler language programmer, since the interface is translated by the macro processor of the assembler. There is also a preprocessor that makes the macro level interface available to the COBOL or PL/I programmer.

- The Command-level interface

  This is available to the COBOL and PL/I programmer. A preprocessor translates the CICS/VS commands to the host language.

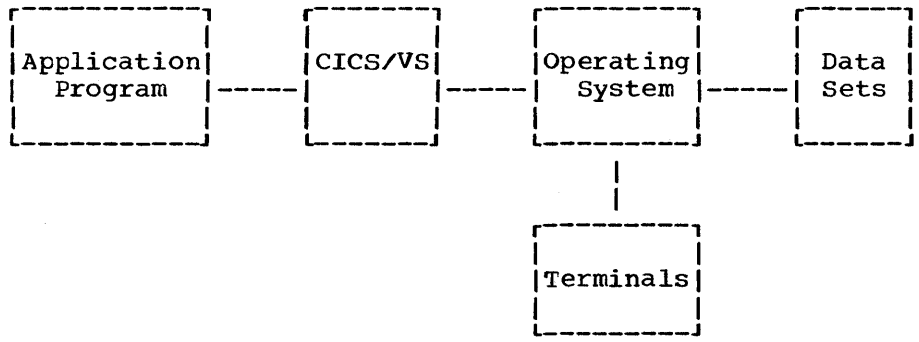

Figure 4-2. CICS/VS in Context

The macro level application interface may be divided into control blocks and function calls. Control blocks are mapped using dummy control sections (DSECTs) with the various fields of the control blocks defined. These DSECTs are brought into the application program by the Assembler COPY command.

Function calls are generated by the expansions of the CICS/VS macros. A typical CICS/VS macro is DFHJC, which invokes the services of Journal Management. Suppose the application programmer codes:

```
DFHJC TYPE=WRITE,
      STARTIO=NO,
      JFILEID=14,
      JTYPEID=3333,
      JCDADDR=JREC,
      JCDLGTH=LREC,
      PFXADDR=JPFX,
      PFXLGTH=LPFX,
      NORESP=OK3A
```

The expansion produced by the macro could be:

```
*********************************************************************
       DS    0H ..... J O U R N A L    C O N T R O L    C A L L .....
       MVC   JCATR2(2),=AL2(1025)     . SET TYPE REQUEST IN JCA      @
       MVI   JCAFID,14              SET JOURNAL FILE ID IN JCA       @
       MVC   JCAJRTID,=H'13107'      JOURNAL-RECORD TYPE-ID TO JCA   @
       LA    14,JREC                GET ADDRESS OF USER-DATA         @
       ST    14,JCAADATA            AND STORE IN JCA                 @
       MVC   JCALDATA,=AL2(LREC)      .USER-DATA LENGTH TO JCA       @
       LA    14,JPFX                GET ADDRESS OF USER-PREFIX       @
       ST    14,JCAAPRFX            AND STORE IN JCA                 @
       MVC   JCALPRFX,=AL2(LPFX)      .USER-PREFIX LENGTH TO JCA     @
       L     14,CSAOPFLA            FROM OPTIONAL FEATURES LIST      @
       USING CSAOPFL,14             GET THE ...                      @
       L     14,CSAJCNA2            ...SECONDARY JCP ENTRY ADDRESS,  @
       DROP  14                     AND THEN ...                     @
       BALR  14,14                  LINK TO PROCESS JOURNAL REQUEST  @
       CLI   JCAJCRC,JCARCNR        Q.IS RETURN-CODE GOOD(NORESP)?   @
       BE    OK3A                       YES - BRANCH                @
*,    * COND KEYWORD ABSENT- DEFAULT (NO) ASSUMED *
*********************************************************************
```

This is a typical expansion. Fields in a control block, in this case the JCA, are set up according to the parameters of the macro, and a call is made to an address held in a field in the CSA. After the call there is a test of the value of a return code that has been set up in the JCA.

The expansions of CICS/VS macros assume that adequate addressibility is provided, so that all the references to control blocks assemble without error. The only immediate analysis of the requests, apart from the different data set up by different macro calls, is the selection of a CICS/VS entry point, there being usually one entry point corresponding to each macro. The entry point addresses are held in the CSA.

Further analysis of the macro invocation takes places within the CICS/VS routine. For instance, in the example just given Journal Management starts, after some housekeeping, by detecting a WRITE request and branching to the appropriate routine within the Journal Management module.

It should be noticed that in some cases the information that needs to be passed to CICS/VS cannot be given entirely by the macros; some of it must be passed through fields in control blocks, which are examined by the CICS/VS routine called. It is then necessary for the application programmer to supplement the macro by setting one or more fields before the macro is issued.

When the source language is COBOL or PL/I, and the Command language translator is being used, CICS/VS commands are expanded into calls from the code generated by the COBOL or PL/I compiler to an interface program. This sets up control blocks according to the command parameters, and hands control to CICS/VS. The command-level interface is complete, so that the application programmer does not need to set fields in control blocks in addition to issuing the commands.

INTERMODULE COMMUNICATION

CICS/VS modules communicate with one another, where possible, by using the CICS/VS application program interface. This is appropriate for calls to those modules, such as Task Management, which provide services to application programs, and many CICS/VS macros have extensions so that they can be used to call for services which are internal to CICS/VS.

Where a call or transfer of control cannot be fitted directly to a macro, Program Management may be invoked (using DFHPC) in the same way as it is used to effect transfers between application programs.

There are no parameter lists in calls and transfers between CICS/VS modules. Parameters are implicit, being fields within the CICS/VS tables and work areas.

CONTROL BLOCKS

COMMON SYSTEM AREA (CSA)

The Common System Area (CSA) is the major CICS/VS control block. It is an area in main storage which exists within the system from initialization time until the CICS/VS is closed down. The CSA contains:

- Register save area
- Pointers to the CICS/VS management modules, control information
- System constants
- Time-management storage
- Work area for statistics
- Task abnormal termination interface
- Pointers to CICS/VS system tables
- Optional user work area

The user work area is available to any task while it has control of the system (that is, for operations performed between requests to CICS/VS).

The CSA is normally addressed by register 13. However, if at the time of a dump a COBOL or PL/I program is being executed then register 13 will point at that program's save area. In this case the CSA can be located by chaining upwards through the save areas.

DISPATCH CONTROL AREA (DCA)

A Dispatch Control Area is created for each task by CICS/VS. It is placed on a DCA chain and used to control task dispatching. It holds a record of the dispatching priority and current status of the task. Two chains of DCAs are maintained, an active chain and a suspended chain. A task is suspended and its DCA placed on the suspended chain when it is expected not to require processing for a long time (for example, when it is waiting for terminal input). A task is active if it is waiting for CPU processing or disk I/O. These chains are addressed through the CSA. Storage for the DCA is acquired by Task Management (using Storage Management) upon task initiation, and released when the task is terminated. The DCA is logically part of the TCA but it is kept separately in order to avoid page faults.

TASK CONTROL AREA (TCA)

A Task Control Area (TCA) is created for each task that currently exists within CICS/VS. Its contents are organized into three logical sections:

- CICS/VS system control section
- Application program communication section
- Transaction Work Area (TWA), which is optional.

The CICS/VS system control section contains the addresses and data necessary for CICS/VS to control a task. Access to data in this area is limited to CICS/VS management programs, CICS/VS service programs, and user-written service programs. Thus the first 128 bytes of the TCA precede the portion of the TCA that a user-written application program sees.

The application program communication section is used for communication between the task and CICS/VS management programs and service programs. Access to this section is provided to both CICS/VS and user-written application programs. It addressed through register 12, and is 256 bytes long.

The TWASIZE operand of the DFHPCT macro instruction (used to define the transaction in the Program Control Table) determines whether a TWA is appended to the TCA. The size of the TWA (if there is one) is specified in this operand. The TWA is acquired at task initiation as part of the TCA and has the same base register as the TCA. The TWA provides the user-written program with unique storage for the duration of the task. This area can be used to pass data or address constants from one program to another within one task. It is used if parameters are passed to a program at a higher logical level. The TWA is reserved for the exclusive use of the application program.

A TCA is created whenever a task is attached. The pointer to the TCA always addresses the application program communication section of the TCA.

Information about Terminal Management macro requests issued by, or on behalf of, the task is copied from the TCA to the Terminal Control Table terminal entry (TCTTE) for the terminal being used with the task. The Facility Control Area Address in the TCA points to this TCTTE. Terminal Management provides services as a separate task initiated by Task Management. A separate TCA is set up for this task, so that it can request services of CICS/VS modules, just as other tasks do. Neither the Terminal Management task nor the Task Management task is simply branched-to by tasks seeking services.Terminal I/O areas are chained off the TCTTE, whereas all other task storage is chained off the TCA. In this way all non-terminal storage can be returned to Storage Management as soon as the task ends.

AUTOMATIC INITIATE DESCRIPTOR (AID)

Automatic task initiation is possible under CICS/VS. An automatically initiated task may be associated with a terminal. CICS/VS employs a queuing technique in order to initiate a task when its terminal destination becomes available. An Automatic Initiate Descriptor (AID) is created for each request for automatic task initiation and added to the chain of AIDs. This chain is sequenced by symbolic transaction

identification within symbolic terminal identification. The CSA contains the address of the first entry of the AID chain.

When an AID is added to the chain, Task Management advises Terminal Management that an automatically initiated task is pending on a particular terminal. This is done by setting an indicator in the associated Terminal Control Table Terminal Entry. Terminal Management advises Task Management when the particular terminal facility is available by issuing the CICS/VS system macro DFHKC TYPE=AVAIL. Task Management initiates the DFHKC TYPE=ATTACH request for the new task.

INTERVAL CONTROL ELEMENT (ICE)

An Interval Control Element (ICE) is created for each time-dependent request received by Time Management. These ICEs are logically chained to the CSA in expiration time-of-day sequence.

Expiration of a time-ordered request is detected by the expired request logic of Time Management, running as a CICS/VS system task whenever the Task Dispatcher gains control. The type of service represented by the expired ICE is initiated, provided that all resources required for the service are available, and the ICE is removed from the chain. If the resources are not available, the ICE remains on the chain and another attempt to initiate the requested service is made when the Task Dispatcher gains control.

Time Management passes expired Interval Control Elements (ICEs) that represent, either time-ordered task initiation requests, or time ordered data records. Task Management treats these as AIDs. If a time-ordered data record has been retained for the new task, the AID remains on the chain until the time-initiated task issues a request (DFHIC TYPE=GET) for the data record, or terminates. The AID is removed from the chain at the time the task is initiated, if no time-ordered data record was associated with the original request.

JOURNAL CONTROL AREA (JCA)

The JCA comprises three major areas: the register save area used by Journal Management, the JCA communication area, and the JCA system prefix build area. This control block is the communication vehicle for Journal Management requests

FILE WORK AREA (FWA)

The FWA is the area in which file records are normally passed between CICS/VS and the application program. It is acquired dynamically from main storage by File Management when reading or updating an existing blocked or segmented record, when adding a new record to a file, or when retrieving records using the browse feature.

FILE INPUT/OUTPUT AREA (FIOA)

The FIOA is acquired dynamically from main storage by File Management whenever a request is made for I/O to an ISAM or DAM data set. The data area is used as the true I/O area, to and from which records are written and read. For all ISAM and VSAM operations,except read without update,records are moved between the FIOA and the FWA.

FILE BROWSE WORK AREA (FBWA)

The FBWA is used during a browse operation against an ISAM or a DAM data set to maintain position in the data set. The FBWA is acquired dynamically when a DFHFC TYPE=SETL macro is issued against such a data set. It is released when the browse operation is terminated via a DFHFC TYPE=ESETL.

DEFERRED WORK ELEMENT (DWE)

A Deferred Work Element (DWE) is created and placed on a DWE chain to save information about an uncompleted event that must be completed before task termination. For example, a DFHFC TYPE=GET,TYPOPER=UPDATE macro instruction causes a DWE to be created. The DWE remains on the DWE chain until the update operation is performed, until a sync point is taken, or until task termination if the task terminates without requesting the update operation.

TEMPORARY STORAGE INPUT/OUTPUT AREA (TSIOA)

The TSIOA is chained off the TCA. It can be acquired by the user, or by Temporary Storage Management in response to a GET or GETQ request when no TSDADDR is specified. TSIOAs acquired by, or on behalf of, a user task are normally released by the task.

TERMINAL INPUT/OUTPUT AREA (TIOA)

Terminal Input/Output Areas (TIOAs) are set up by Storage Management and chained to the Terminal Control Table Terminal Entry (TCTTE) as needed for terminal input/output operations. The TCITE contains the address of the first terminal-type storage area obtained for a task (the beginning of the chain) and also the address of the active TIOA.

STORAGE ACCOUNTING AREA (SAA)

A storage area handled by Storage Management has a header, and, possibly, a trailer, which is a copy of the header. The whole area is referred to as a Storage Accounting Area (SAA).

CICS/VS EXECUTION

When a CICS/VS system is in operation many concurrent activities are normally taking place:

- Teleprocessing Network. Of the active terminals, some are transmitting messages to CICS/VS, others are awaiting output messages, and still others are receiving messages. Data is being passed between CICS/VS and peripheral devices in response to I/O requests.

- Peripheral Devices. Some of the active peripheral devices are copying data from the CICS/VS region, others are recording data in the CICS/VS region.

- Central Processing Unit. Many tasks, at various stages of processing, are concurrently resident. Unless CICS/VS has executed an operating system WAIT, either a CICS/VS service module or a CICS/VS task is executing and other tasks are queueing to be dispatched, or waiting for input or output to complete.

Thus, a multitasking teleprocessing system is highly dynamic. The concurrent activities described above compete with each other for system resources: CPU time and main storage; channel, control unit, and storage device services; and communication lines.

TRANSACTIONS AND TASKS

It is desirable to distinguish between transaction and task, as illustrated in Figure 4-3.

Figure 4-3. shows a transaction made up of four major parts:

1. The Input Message. This contains the transaction ID and the data required by the transaction.

2. The Task. This performs the CPU and file processing required by the transaction.

3. The Output Message. This is composed and scheduled for transmission by the task, but is transmitted by Terminal Management.

4. The Definite Response (VTAM only). When this is required, the transaction is incomplete until a definite response is received from the terminal by CICS/VS.

A transaction is the unit of processing initiated by a single request, which may require the initiation of one or more tasks. A task is a concept internal to CICS/VS and is the processing between a CICS/VS attach and a CICS/VS detach.

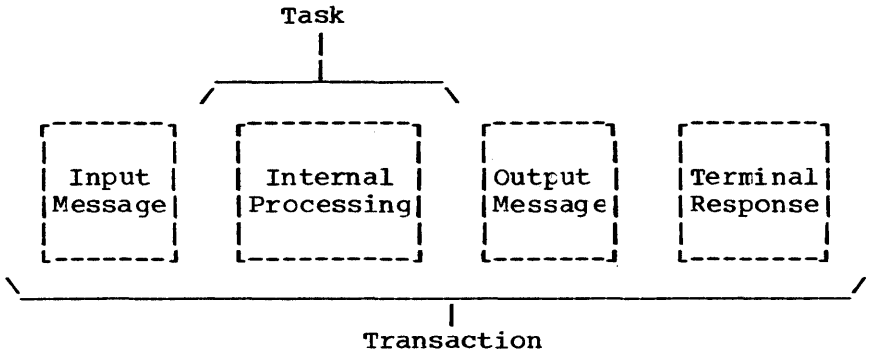

Figure 4-3. Transaction and Task

This distinction between transaction and task is also important in an understanding of the mechanics of enqueuing, dequeuing, logging, and deferred processing.

CICS/VS RECOVERY

Certain general principles govern the way CICS/VS protects resources and transactions:

- The enqueue on a protected resource (a record, a Transient Data destination, or a Temporary Storage data element or queue) precedes execution of the first macro instruction that makes or leads to a modification of the resource (for example, PUT, GET FOR UPDATE, DELETE PUTQ).
- The dequeue does not occur until the end of the task or logical unit of work
- Before a resource is modified, the information needed to back out the modification is written to an external storage device. For example, a GET FOR UPDATE gets and logs the 'before image' of a record to be modified. When the record is subsequently about to be updated by a PUT, the logged 'before' image may still be resident in a journal control buffer. This buffer is written to the log before the PUT is executed. A copy is also retained in a manin storage dynamic buffer for use by Dynamic Transaction Backout.
- Certain irreversible operations (for example, TD PURGE, IS RELEASE or PURGE,or transmission of a protected message) are deferred until the task is complete.

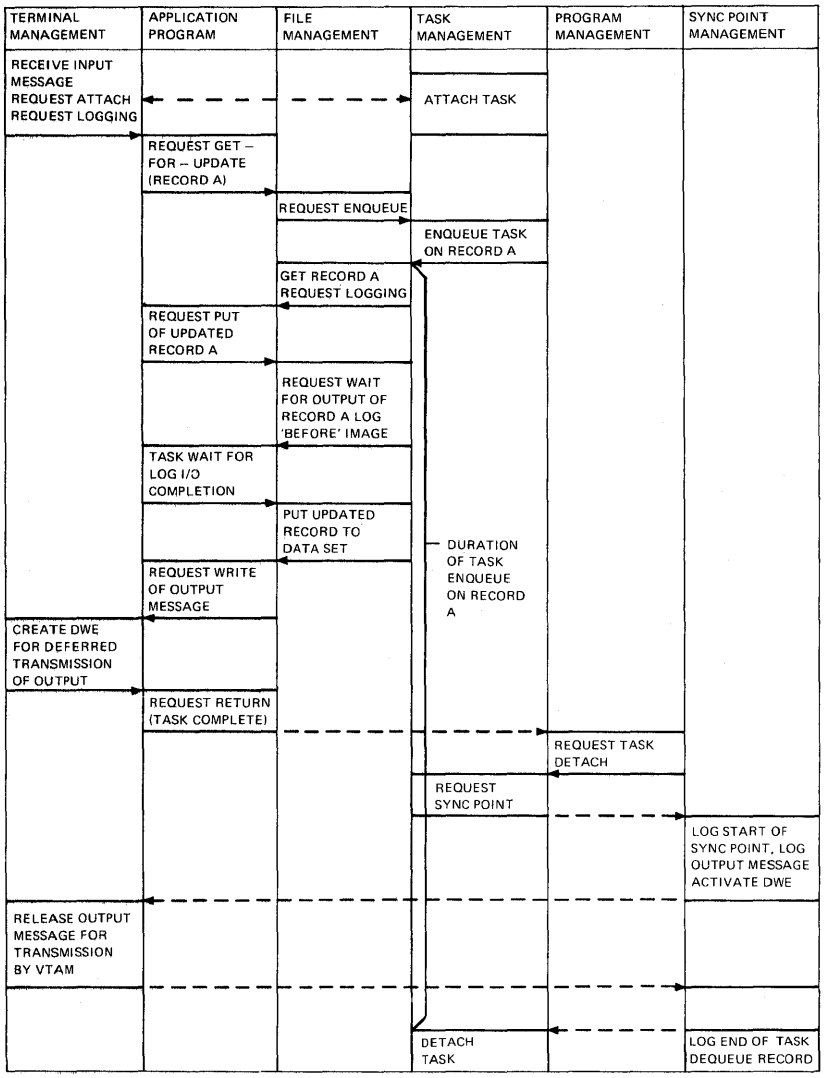

Figure 4-4. CICS/VS Dynamics

In CICS/VS, deferring an operation until completion of a task is done in two steps:

1. A record is made of the function to be performed and the data needed in a Deferred Work Element (DWE).

2. A DWE processor is activated to execute the operation after the task has issued its final RETURN, or a DFHSP.

Each CICS/VS management program that can be invoked for deferrable operations has a subroutine for performing deferred processing. The subroutine operates on any DWEs created during the task. The address of

this subroutine is in the DWE. The DWEs are scanned and processed by CICS/VS immediately before the task is detached.

Figure 4-4. illustrates these ideas by showing the flow of control for a relatively simple transaction with the following characteristics:

- Input and Output messages are protected. For example, they must be logged, and the transmission of the output message must be deferred until the task completes.
- The task updates one record (called 'Record A' in Figure 4-4.) in a protected data set.

The time sequence in Figure 4-4. runs from the top of the diagram to the bottom. Arrows indicate the flow of control from one module to another. Lines with arrows at both ends indicate flow to a module providing a requested service, followed by a return to the module that requested the service.

The solid black line in the File Management column illustrates the duration of the enqueue on Record A.

The legend 'Task wait for log I/O completion' in the Application Program column illustrates the additional wait time imposed on a task by logging requirements. Since Journal Management, which does the logging but is not shown in the diagram, is a CICS/VS subtask, impact of the extra wait on performance is minimal. There may be no task wait time at all, if the journalling activity has already forced the buffer to be written out.

There are important task related activities that occur after a task has issued its final RETURN. The task is not detached until Sync Point Management has activated all deferred processing, logged task completion, and dequeued all resources held by the task.

LONG-RUNNING TASKS, SYNC POINTS, AND LOGICAL UNITS OF WORK

The sync point illustrated in Figure 4-4. is taken as a result of the execution of a sync point macro instruction by Task Management at task detach time.

An application program can also take a sync point by executing a DFHSP TYPE=USER macro instruction.

Issuing a user sync point forces the logging, deferred processing and dequeuing activity that is normally not done until after a task completes (refer to Figure 4.4). User sync points can be used to divide a CICS/VS task into a series of sequential, self contained sections. In CICS/VS each section is called a logical unit of work (LUW).

To minimize resource conflict and delays due to deferred processing, the application program will divide long-running tasks into LUWs, by issuing user sync point macro instructions as frequently as the logic of the application allows.

Sync points also reduce the amount of work that has to be backed out during emergency restart, and the amount of work that has to be redone after the system is restarted.

For backout purposes, taking a user sync point has the same effect as attaching a task. None of the work done by the task before the sync point was taken is backed out during CICS/VS emergency restart. This

should be remembered when using the sync point macro instruction to delimit an LUW.

MULTIPROGAMMING MULTITASKING MULTITHREADING

Within the CICS/VS region, a task executes until it has to wait for the completion of some external event (usually I/O). CICS/VS notes that the task is waiting for something external to happen and schedules another task. Similarly, when all tasks in the CICS/VS region are awaiting I/O completions (or external events for other CICS/VS tasks), CICS/VS issues an operating system WAIT, permitting the activation of another partition that can execute while CICS/VS is idle.

MULTIPROGRAMMING

The CICS/VS system as a whole may share the computer and operating system with batch programs (or even with other DB/DC systems) in other partitions. If CICS/VS is executed in such a multiprogramming environment, it is usually in the highest priority partition (except for operating system components). Batch partitions receive control from the operating system only when CICS/VS has no dispatchable transactions. Thus, as long as there is a transaction ready for processing, CICS/VS maintains system control. Control is released to allow the operating system to continue a job in another partition only when there are no more transactions ready. CICS/VS regains control as soon as any previously waiting CICS/VS transaction is ready to continue, or, if all active transactions are in a wait state, as soon as a new transaction code is entered at a terminal.

MULTITASKING

To achieve its objective of providing fast response to terminal users, CICS/VS executes in a multitasking mode of operation within one partition of main storage. Such multitasking within one partition is analogous to multiprogramming within the total operating system environment. Generally, tasks are initiated as a result of transactions entered at terminals. Whenever one task (transaction) is forced to wait for completion of an I/O operation, availability of a resource, or some other cause of delay, processing of another task within the system is initiated or continued.

In the example in Chapter 1, with only one terminal active, when the File Management GET and Transient Data PUT operations were completed, control was described as returning to the single task. With multiple tasks running, operations involving WAITs result in temporary suspension of the task. The highest priority task that is ready for processing is dispatched. When the inquiry's file read or write operations are completed, control returns to it, according to its priority. Through this task switching, CICS/VS can overlap the prccessing of one task with the I/O operations of others.

MULTITHREADING

In CICS/VS, more than one transaction may require the same program. Rather than have many copies of a program in storage, one copy is used

by many tasks. An application program, especially one with several I/O operations, may have many tasks associated with it, some having gone through the first section of the program up to the first I/O instruction, some through the second section, and so on. Each task awaits its turn to continue through the next part of its program.

To control multithreading, Task Management uses a Task Control Area (TCA) for each task. This allows Task Management to determine where each task is within a program, or where it should return to resume processing, when it receives control.

STORAGE

OPERATING SYSTEM STORAGE

A system such as DOS/VS or OS/VS provides a user (in this case, CICS/DOS/VS or CICS/OS/VS) with more address space than the real storage capacity of the computer on which it is run. This increase is accomplished by means of a paging technique, supported by both the system hardware and the operating system software.

Hardware real storage is formatted into standard-sized units, called page frames. Virtual address space on direct access storage (called the page data set) is divided into units of the same size, called page slots. When an address space is referenced, the virtual address is translated to a real address by the Dynamic Address Translation (DAT) unit in the hardware. The desired address space must occupy a page frame in real storage. If it does not, a page-fault interrupt occurs, and control is passed to the operating system's page management logic. The page manager selects a real storage page frame that can be used for the referenced virtual address space (page). When nearly all page frames have been filled, a page replacement operation is required. The operating system replaces first those pages that have not been referred to for the longest period of time. If a page to be replaced has been modified, that page must be written out onto the page data set before the new page can be read in. The real storage page frame freed in this way is then used as the referenced page.

CICS/VS ADDRESS SPACE

The CICS/VS main storage address space is structured as follows:

- Virtual Storage area specified by the DOS/VS or CS/VS operator starting up CICS/VS
- CICS/VS nucleus, comprised of CICS/VS modules, CICS/VS system tables, and program storage for resident (fixed and non-fixed) application programs
- CICS/VS dynamic storage used for CICS/VS control areas, input/output areas, work areas, and program storage for nonresident loaded application programs

The storage organization for CICS/DOS/VS is shown in Figure 4-5.

SUBPOOL ALLOCATION OF DYNAMIC STORAGE

Storage Management controls CICS/VS dynamic storage space. It allocates storage, dependent on the type of request, from different subpools. Each subpool obtains and returns storage in page multiples from the dynamic storage area, which is described in the CICS/VS Page Allocation Map (PAM). The address of the PAM is kept in the CSA.

The subpools are:

- Mixed
- Isolated
- Program
- Control
- Teleprocessing
- RPL

During system preparation, the CICS/VS user can identify certain application transactions as short-term and others as long-term. These specifications affect the way in which CICS/VS allocates storage for the transactions. Storage is allocated when it is needed by concurrently

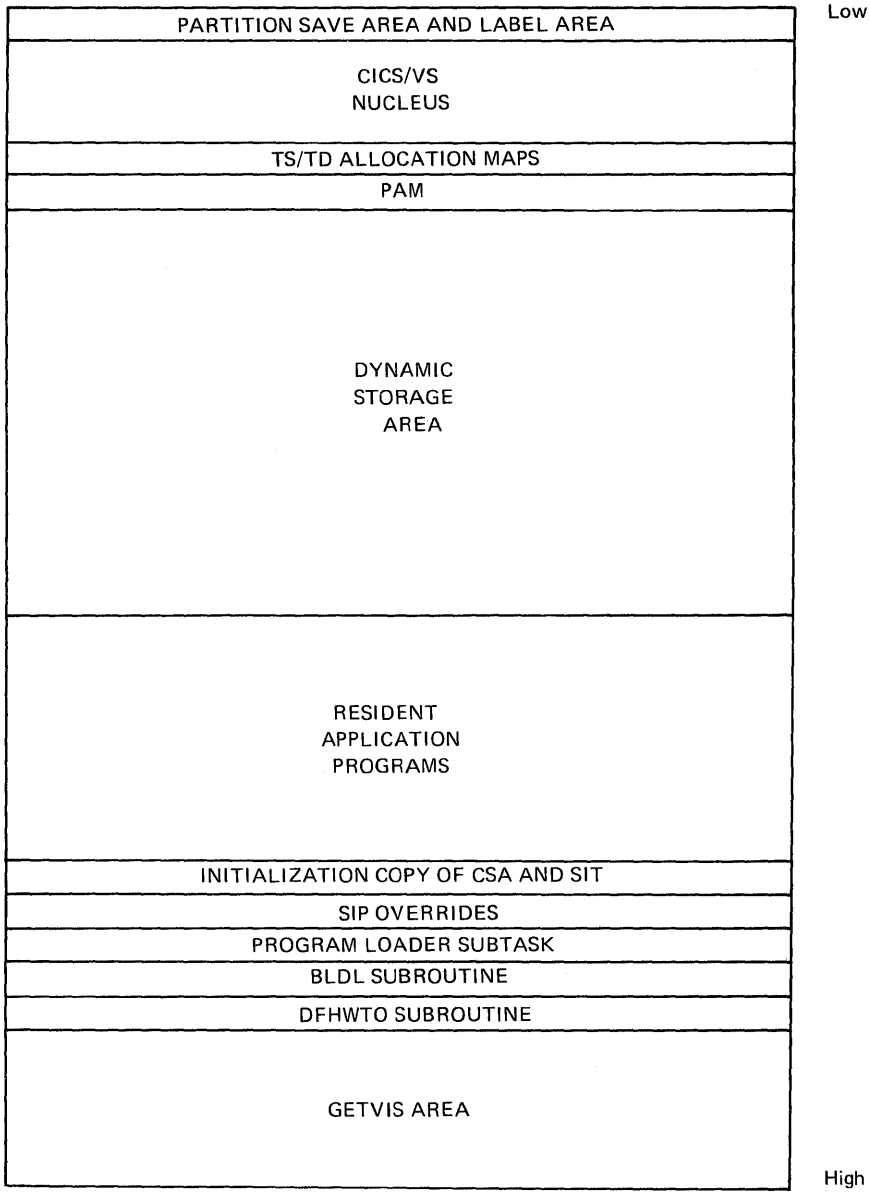

Figure 4-5. Storage Organization under CICS/DOS/VS

executing short-term transactions from a particular set of page frames (the mixed subpool). Thus, this subpool contains a mixture of storage for all the short-term transactions executing at a particular point in time, and is heavily used. As this short-term storage is freed, and eventually, as all of the storage in a page frame is freed, that storage is returned from the mixed subpool to a common available page pool for reallocation to the mixed subpool, or other subpools when needed.

On the other hand, CICS/VS Storage Management allocates all storage requests for each long-term transaction in a separate page frame. These pages are in the isolated subpool and undergo short periods of high

activity followed by long periods of inactivity (such as waiting for a terminal operator to enter data for a transaction).

All dynamically loaded application programs are allocated contiguous full page frames in the program subpool. Any program less than one full page (2K or 4K, depending on the operating system) in size is allocated one whole page. Any program requiring one and a half pages is allocated two pages, and so on.

The control subpool is used for storage needed by CICS/VS to manage its own resources, such as the space needed to record information about transactions to be initiated at some future time, and space for tasks which are waiting on the availability of a particular resource. A teleprocessing subpool is used for line and terminal input/output areas.

Transactions can be designated as primed, in which case some Storage Management requests are satisfied from a block of storage obtained at task initiation. This storage is held at the end of the transaction for re-use by a new transcation of the same type. This scheme speeds up storage requests.

DATA SETS

SYSTEM DATA SETS

The access methods for use with CICS/VS system data sets and the record format of these data sets are predefined within CICS/VS. Any required formatting of the data sets is performed by CICS/VS during System Initialization or by the maintenance functions. The system data sets are:

- CICS/VS Program Library
- Restart data set
- Dump data set
- Intrapartition data set
- Temporary Storage data set
- System Log data set
- Automatic Statistics data set
- Auxiliary Trace data set

System data sets other than the Dump data set and System Log must be located on direct access storage. The Dump data set and System Log may be on either direct access storage or magnetic tape. Whether all of the data sets are required, and the amount of space needed for each, depends on the CICS/VS options selected at CICS/VS system generation and the extent to which they are used.

CICS/VS Program Library

The CICS/VS Program Library contains all user-written programs and CICS/VS programs to be loaded and executed as part of the online system. This group includes the control system itself and certain user-defined system control tables essential to CICS/VS operation. The library contains program text and, where applicable, a relocation dictionary for a program. The contents of this library are loaded asynchronously into CICS/VS dynamic storage for online execution.

Restart Data Set

The Restart data set is a BDAM file used by the CICS/VS Keypoint program (DFHKPP) to save certain system environment information at system termination so that CICS/VS can be warm started later if desired. This optional facility can be invoked to warm start the following CICS/VS control information:

- Program Control Table (PCT)
- Processing Program Table (PPT)
- Terminal entries (nonswitched)
- File Control Table (FCT)
- Common System Area (CSA)
- CSA Optional Features List
- Destination Control Table (DCT)
- Transient Data intrapartition space allocation bit map
- IDs and RBAs for Temporary Storage auxiliary destinations and queues
- Temporary Storage space allocation bit map
- Interval Control Elements (ICEs) and Automatic Initiate Descriptors (AIDs)
- Batch Control Areas (BCAs) and Write Request Elements (WREs)

Dump Data Set

The Dump data set is used by Dump Management to record dumps of transactions (tasks) within the system. An optional data set, it is sequential in organization and can be formatted and printed by the CICS/VS Dump Utility program. If the user desires, he can define two dump data sets (DFHDMPA and DFHDMPB), alternating between them during online execution of CICS/VS.

Intrapartition Data Set

The Intrapartition data set is an optional data set that can be used for the queuing of user data and CICS/VS data by Transient Data Management. The data is stored chronologically into this data set, by symbolic destination. Such data can be retrieved or routed to other destinations, and space within the data set can be reused.

Temporary Storage Data Set

The Temporary Storage data set is an optional data set, required only when the general-purpose scratch pad or queuing facilities of Temporary Storage Management are utilized, if the time-ordered automatic task initiation feature of CICS/VS is generated, or if the paging or routing facility of Basic Mapping Support or Message Switching is used. The data set is a VSAM entry-sequenced data set with variable-length records within fixed-length control intervals. User data is stored into this data set under a dynamically provided symbolic identification for subsequent retrieval and release.

System Log Data Set

The System Log data set is used for logging activity on protected resources. During an emergency restart, the Recovery Utility program reads the System Log backwards to retrieve the information needed to restore system activity tables and to retrieve transaction backout data. To restore the system activity tables, at least one activity keypoint must be present on the System Log. Transaction backout data is that data logged by tasks that did not complete processing before termination

time. All activity of the task, or all activity of the current logical unit of work (LUW) within the task, will be backed out.

Automatic Statistics Data Set

The automatic statistics data set is used by the automatic statistics summarization control program (DFHSTSP) to record system statistics. The data set is sequential in organization and located on either tape or disk (DFHSTN or DFHSTM, respectively). The data set is created using extrapartition transient data services and is read by the Automatic Statistics Utility program (DFHSTUP) to create the statistics report.

Auxiliary Trace Data Set The auxiliary trace data set is used by Trace Management to record all trace entries that occur when the auxiliary trace function is active. The data set is optional and is sequential in organization. The Trace Utility program (DFHTUP) can be used to print records from this data set.

USER DATA SETS

User data sets comprise those data sets that form the CICS/VS user data base and Transient Data extrapartition data sets, containing data coming into or going out of the data base/data communication environment. They may also include Terminal Management sequential data sets, Data Language/I data sets, and journal data sets.

Data Base Data Sets

User data base data sets must be placed on direct access storage and can be accessed under control of DAM (on DOS), or BDAM, ISAM, or VSAM (on OS). Under any of these access methods, the user has a great deal of flexibility in defining the structure of the data sets. Indexing may be used to achieve indirect addressing of data or to define segmented records to promote efficient handling of data, or to provide for potential saving of disk storage space.

Transient Data Extrapartition Data Sets

Extrapartition data is usually routed to or from high-speed input/output devices and typically consists of blocked, variable-length records. The extrapartition disposition facility is intended for use with data collection and data entry applications, and for output of data to be subjected to subsequent (usually offline) processing.

Terminal Management Sequential Data Sets These data sets are created by entries for other than a telecommunication device in the Terminal Control Table. The data sets must be unblocked and are normally sequential data sets on disk or tape but may be on a card reader or printer.

Data Language/I Data Sets

On OS/VS Data Language/I data sets are accessed by means of the Data Language/I (DL/I) facility of the IBM Information Management System/Virtual Storage (IMS/VS). Such access requires the installation of the IMS/VS Data Base System (5740-XX2).

On DOS/VS the DL/I access method (5746-XX1) is used.

Journal Data Sets

If CICS/VS Journal Management is used, entries in the CICS/VS Journal Control Table (JCT) identify user journal data sets. These data sets may be on either tape or disk. Each data set must be given a journal file identification.

Chapter 5. CICS/VS Advanced Communication Systems

This chapter provides an overview of the operation of CICS/VS when it is being used to communicate with a subsystem.

Communication subsystems that can be used through CICS/VS include:

- The IBM 3600 Finance Communication System
- The IBM 3650 Retail Store System
- The IBM 3790 Communication System
- The IBM 3270 Information Display System
- The IBM 3767 Communication Terminal
- The IBM 3770 Data Communication System

A CICS/VS advanced communication system is a teleprocessing system that includes both CICS/VS and one of the above mentioned subsystems. A CICS/VS advanced communication system requires the following components:

- CICS/VS and its application programs in the host CPU
- A telecommunications access method (VTAM, TCAM, or EXTM)
- An operating system (DOS/VS, OS/VS1, or OS/VS2)
- Unless the subsystem is locally connected, one or more IBM 3704 and/or 3705 Communications Controllers and Network Control Programs/Virtual Storage (NCP/VS)
- One or more advanced communication subsystems, including at least one terminal for the control operator, and possibly a subsystem application program

When requesting communication with terminals in an advanced communication subsystem, CICS/VS application programs use the same macro instructions that are used to communicate with any other terminal. The difference is that requests involving an advanced communication subsystem are sent through a communications controller and the CICS/VS application program communicates with logical units (which include application programs, control programs, or terminals).

CICS/VS application programs can use versions of the DFHTC macro instruction to transfer data to and from the logical units. As with BTAM-supported terminals, data is transferred between the CICS/VS application programs and CICS/VS, by means of a Terminal I/O Area (TIOA).

A logical unit need not correspond physically with a terminal, but may consist of several different devices. Logical units are identified to CICS/VS by DFHTCT macro instructions in the same way as start-stop and binary synchronous communication (BSC) terminals are identified.

Some subsystems are programmable, and the logical units may correspond to subsystem application programs. These subsystem application programs communicate with CICS/VS application programs, and in this case the characteristics of the logical unit in the subsystem,

as it appears to CICS/VS, will depend on the way the corresponding subsystem application program is written.

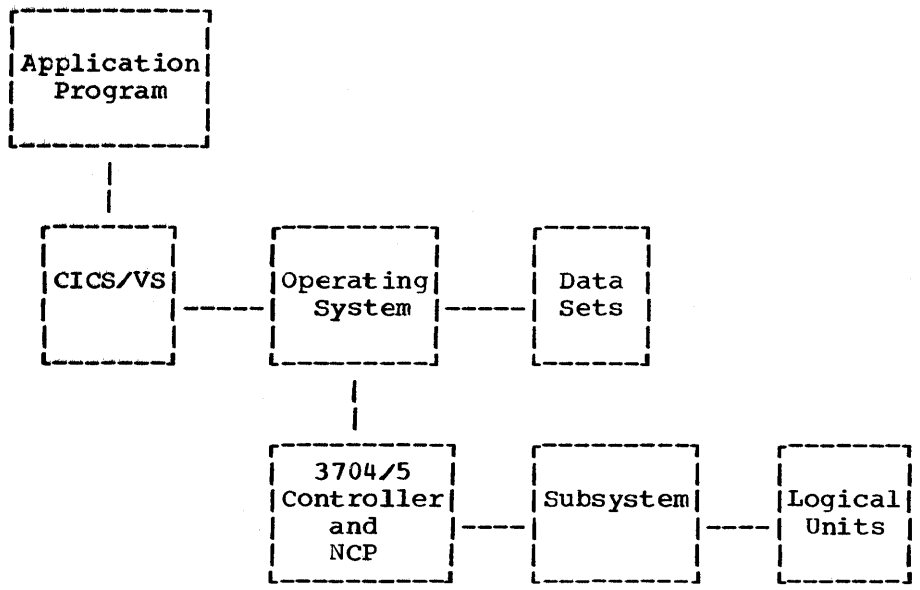

Figure 5-1. CICS/VS in Context

The basic operation of a CICS/VS advanced communication system involves:

- Establishing connections between CICS/VS and logical units
- Sending data between CICS/VS application programs and these logical units
- Terminating connections between CICS/VS and logical units

In the typical case, where VTAM is used, the CICS/VS advanced communication system is started in the following steps:

1. The operating system of the host CPU is loaded into the system.
2. VTAM is started. Depending on options specified, starting VTAM can include activating any or all of the following: NCP/VS, advanced communication subsystems, and CICS/VS. These components can also be activated individually by the VTAM network operator.
3. The SDLC cluster controller is initialized for the advanced communication subsystem. This initialization causes the designated operating environment to be loaded into the SDLC cluster controller.
4. CICS/VS is started. Note that CICS/VS must be started (like other problem programs) in addition to being activated by VTAM. Once started and activated CICS/VS can then connect CICS/VS application programs and logical units and can then enable these nodes to communicate with each other.

Figure 5-1. shows the interconnections between the components of an advanced communication system.

SESSIONS

Before communication can take place between a CICS/VS and a logical unit, a session must be established connecting the logical unit with the CICS/VS application program.

A logon is the only method by which a session between a logical unit and CICS/VS can be initiated. Logon may be initiated by the logical unit, by the network operator, by the CICS/VS master-terminal operator, automatically by the access method, or by CICS/VS itself. CICS/VS uses a simulated logon macro instruction to initiate a logon on behalf of logical units such as those that:

- Contain output-only terminals
- Are defined by the user as secure terminals to which access is controlled by CICS/VS

Logical units for which CICS/VS is to initiate logons are identified to CICS/VS through the Terminal Control Table (TCT). CICS/VS also uses the SIMLOGON macro instruction to obtain logical units that are requested by the master-terminal operator or that are involved in an automatic task initiation (ATI) but are not currently connected.

Once a logical unit is connected to CICS/VS, it remains connected until:

- It is reallocated by the access method.
- The logical unit itself requests disconnection.
- CICS/VS, the access method, the NCP/VS, the logical unit, or the entire system is deactivated.
- The CICS/VS application program requests disconnection.

CICS/VS uses the access method close destination macro instruction to disconnect logical units.

INITIATING COMMUNICATION

When the subsystem receives a command to open the session, it allocates a session to the logical unit and transmits a connection request to CICS/VS. (In the subsystem, a session is treated as a resource; the maximum number of sessions available to a logical unit is specified when the subsystem is generated). When the logical unit sends data, the subsystem control program transmits the designated message to the host.

When CICS/VS receives the connection request and the first message from a logical unit it initiates a transaction. The transaction initiated depends upon the transaction identification. If the TRANSID operand is specified in the DFHTCT TYPE=TERMINAL macro instruction for the logical unit, that identification is used; otherwise, the identification of the transaction must be inserted in the first four characters of the message transmitted by the logical unit.

The initial request to put data must also include a begin bracket designation. Bracket protocol is used for each transaction between the CICS/VS application program and the logical unit; this protocol delimits the CICS/VS transaction.

Once in session with CICS/VS, the logical unit can initiate any number of CICS/VS transactions. These subsequent transactions are also initiated whenever the logical unit sends data with a begin bracket designation.

TERMINATING COMMUNICATION

A session is terminated by removing the logical connection between a logical unit in the subsystem and CICS/VS. There are two types of session termination: orderly and immediate.

Orderly Termination

An orderly termination occurs when the logical unit is allowed to complete any transactions currently in progress before the session is terminated.

If the session is in a bracket state, CICS/VS must issue an end bracket. The logical unit then issues a session-end request and then, if the session end has not completed, the logical unit should issue a get-status wait.

When CICS/VS receives the session end request, it issues a CLSDST macro instruction for the logical unit causing the access method to send the clear indicator followed by the unbind-session indicator.

CICS/VS initiates orderly termination by issuing the shutdown indicator. The subsystem recognizes the shutdown indicator and returns a signal to CICS/VS. CICS/VS then issues a CLSDST macro instruction for the logical unit causing the access method to send the clear indicator followed by the unbind-session indicator.

Immediate Termination

Immediate session termination is unconditional and ignores outstanding transactions or the processing state of the logical unit. CICS/VS flags immediate session terminations as abnormal.

A request for immediate termination may be initiated by:

- The logical unit closing the session
- CICS/VS automatically issuing a CLSDST for the logical unit

While processing transactions, the logical unit may encounter a condition, such as a program check, which precludes further transaction processing. In such cases, the subsystem controller should issue the a terminate indicator to CICS/VS.

CICS/VS terminates a session immediately by issuing a CLSDST for the logical unit. No warning is given to the logical unit, which cannot stop the termination of the session. The subsystem does not transmit data to CICS/VS on behalf of the logical unit after receiving the clear-unbind sequence.

SIGN-OFF

Terminal operator sign-off using the CSSF message does not cause session termination with CICS/VS. The operator identification and security key is removed from CICS/VS, but the logical unit remains in session capable of sending or receiving data.

Sign-off using CSSF GOODNIGHT causes session termination. CICS/VS places the logical unit in receive-only mode and initiates an orderly session termination.

DATA TRANSMISSION

While a transaction is in progress, the CICS/VS application program uses CICS/VS macro instructions to request CICS/VS services (including data transfer services for the logical unit). CICS/VS, in turn, uses access method record-mode macro instructions to request the data transfer operations. The access method adds routing information to the data stream and uses the facilities of the operating system and NCP/VS to transmit the requests to the appropriate logical unit. Finally, the advanced communication application program for the designated logical unit uses subsystem instructions to request the services of the advanced communication subsystem.

Terminal services provides a basic method of sending and receiving data during communication with subsystem terminals. Operands and parameters are added to the DFHTC Terminal Management macro instruction to provide a means of using the facilities of VTAM and the subsystem.

Once a session is established, the format and content of the data are the responsibility of the CICS/VS application program and the logical unit. All data exchanged is in a user-defined format.

READING DATA FROM A LOGICAL UNIT

The DFHTC TYPE=READ or CONVERSE macro instruction is the only method available for reading data from a logical unit.

The IOTYPE operand of the DFHTC TYPE=READ macro instruction assists the application program in overlapping and synchronizing processing with I/O operations.

When a task is attached by CICS/VS, the input data is presented to the CICS/VS application program. The data, whether it be from a start-stop or a BSC terminal or from a logical unit, is presented to the CICS/VS application program in a terminal I/O area (TIOA) addressed through the TCTTE.

When a CICS/VS application program issues a DFHTC TYPE=READ macro instruction to obtain input from a logical unit, the resultant data is placed in a TIOA. Save requests for reads from logical units are also honored by CICS/VS. A read request from a CICS/VS application program is not completed (that is, no assumptions should be made concerning input data in the TIOA) until the application program issues a wait.

The IOTYPE option of the DFHTC TYPE=READ macro instruction permits the CICS/VS application program to separate read and wait requests and

to specify whether the actual input operation should be initiated before or after a wait is requested.

If IOTYPE=IMMED is specified, CICS/VS initiates the read operation immediately, regardless of whether a wait accompanied the read request. Once the read is initiated (that is, the input request is sent to VTAM), control is returned to the CICS/VS application program if a wait is not specified. The application program is thus able to continue processing while other components in the system are processing the read. When the CICS/VS application program needs the data from the read, it issues a terminal-control wait to ensure that the data has arrived in the TIOA.

If IOTYPE=DELAY is specified, CICS/VS does not initiate a read until a wait request is issued or the transaction releases control.

The default for a read request is IOTYPE=DELAY; this default processing is compatible with the read processing for start-stop and BSC terminals. The wait can accompany the read request, or it can follow the request at a later point. In either case, the read operation does not begin until the wait is issued or the transaction releases control, and the CICS/VS application program must wait while the operation completes.

Synchronizing Logical-Unit Input Operation

A wait request must be issued by the CICS/VS application program before the input data can be processed by the application. A wait request ensures that the data is in the TIOA. For an IOTYPE=IMMED, the data could have arrived prior to the wait request. For an ICTYPE=DELAY, the wait causes the input operation to be initiated; the data is available only after the wait is satisfied.

Unsolicited Input

If a task is in progress and unexpected data (that is, data from a terminal for which a read request is not issued) arrives from a start-stop or BSC terminal, CICS/VS may ignore the data, which will then be lost. In contrast, if unexpected data arrives from a logical unit, it is queued and is used to satisfy any future input requests for the logical unit.

Inbound Function Management Header (FMH)

The CICS/VS application program can request notification when a function management header (FMH) is included in the data received during a read from a logical unit. The FMH is a variable length field that can be sent from the logical unit via the TIOA; when present, the FMH is at the front of the TIOA.

The FMH may be any length up to 256 bytes, its first byte contains its length; its second byte contains a type code. The system programmer specifies in the PCT whether or not inbound FMHs will be passed to the application program. He can specify that no inbound FMHs will be passed, that only the FMH at the end of the data set will be passed, that all inbound FMHs will be passed, or that FMHs are to be processed by the data interchange program. If he specifies that all inbound FMHs will be passed to the application program, he should code the INBFMH operand of the DFHTC TYPE=READ or WAIT macro instruction. This operand will instruct CICS/VS to give control to a user-written routine whenever an inbound FMH is received.

When input data is received as a chain of request/response units (RUs), only the first (or only) RU of the chain is preceded by an FMH.

Chain Assembly

The system programmer can specify, in the TCTTE, whether or not chain assembly is to occur. If chain assembly is to occur, then instead of each read request being satisfied by one RU, until the chain is complete, the whole chain is assembled and sent to the CICS/VS application program in a single TIOA, satisfying just one read request. This ensures that the integrity of the whole chain is known before it is presented to the application program. If the EOC operand is specified, the end-of-chain routine will receive control for every read request (except at the end of the data set).

WRITING DATA TO A LOGICAL UNIT

Although the CICS/VS application program may be communicating with a logical unit instead of with a terminal, it still uses the DFHTC TYPE=WRITE macro instruction to transmit output data to its destination. When issuing a write request, TCTTEDA must point to the TIOA containing the output data, and TIOATDL must specify the length of the data including the three or more bytes for the function management header (FMH).

Conversational Write

The macro instruction DFHTC TYPE=CONVERSE expands into WRITE, WAIT followed by a READ, WAIT for a logical unit. Data to satisfy the write request must have been set up in the TIOA; the data resulting from the read is placed in a TIOA by CICS/VS.

Control is returned to the CICS/VS application program following the completion of the read-wait sequence. The address of the input data is found in TCTTEDA; the length of the input data is specified by TIOATDL.

Overlapping Logical-Unit Output Operations

Write operations can overlap other processing performed by a CICS/VS application program, just as read operations can. Thus, if IOTYPE=IMMED is specified for a write request, the request is initiated by CICS/VS, and control is returned to the CICS/VS application program. The application program can continue processing while the write operation is being processed by other components in the system. A wait request can be issued by the application program when it nedds to verify that the write has completed.

If IOTYPE=DELAY is specified, the write is not initiated until a wait is requested, a sync point is requested via a DFHSP macro instruction, the task releases control, or the task terminates. As with the read request, IOTYPE=DELAY is the default in order to remain compatible with writes for start-stop and BSC terminals.

Synchronizing Logical-Unit Output Operations

Write requests can be synchronized with other processing done by CICS/VS application programs, just as read requests can be. Thus, a wait

request must be issued to ensure that a write with IOTYPE=IMMED has completed or to complete a write request with IOTYPE=DELAY.

If IOTYPE=IMMED is specified, the output operation begins immediately, and parameters such as SAVE must not be added tc the write request by another DFHTC request. The wait completes after the access method has accepted the output request.

Chaining of Output Data

As with input data, output data is transmitted as request/response units (RUs). If the length of the data supplied in the TIOA by the application program by means of a DFHTC TYPE=WRITE macro exceeds the RU size, CICS/VS automatically breaks up the data into RUs and transmits these RUs as a chain. During transmission from CICS/VS to the logical unit, the RUs are marked FOC (first-), MOC (middle-), or EOC (end-of-chain) to denote their position in the chain. An RU that is the only one in a chain is marked OC (only-in-chain).

The system programmer can specify, by using the CCONTR option of the OPTGRP operand of the DFHPCT macro instruction, that the application program can control the chaining of outbound data. If CCONTR is specified, the application program can inhibit the end-of-chain marker on the last (or only) RU resulting from the write request. Ihe data supplied in the TIOA for the next write request will then be treated as a continuation of the chain. To accomplish this chain-building function, the write request must include the CCOMPL=NC operand (specifying that the chain is not yet complete).

Function Management Header (FMH)

When a message is transmitted to a logical unit, it may start with an FMH. The FMH can be built either by CICS/VS or by the CICS/VS application program. If built by CICS/VS, the CICS/VS application program must be sure to reserve the first three bytes of the message for the FMH.

If CICS/VS is to build the FMH, a write request specifies or defaults to FMH=NO. If the application program builds its own FMH, FMH=YES must be specified in the write request.

Bracket Protocol

Bracket protocol may be used when CICS/VS communicates with a subsystem logical unit. For the most part the use of brackets is transparent to the CICS/VS application program.

Only on the last write operation to a logical unit does the bracket protocol become apparent to the CICS/VS application program. On the last output request to a logical unit, the CICS/VS application program may specify LAST in the DFHTC TYPE=WRITE macro instruction. The LAST specification causes CICS/VS to transmit an end-bracket indicator with the final output message to the logical unit. This indicator notifies the subsystem logical unit that the current transaction is ending.

If the LAST operand is not specified, CICS/VS waits until the task detaches before sending the end-bracket indicator.

DATA CHAINING

During system generation, CICS/VS is informed of the maximum permitted data length for a single outbound transmission to the logical unit. If a CICS/VS application program sends a message longer than the maximum permitted length, CICS/VS automatically divides the message and sends it as a chain of transmissions.

CICS/VS sends a cancel indicator whenever the logical unit returns an exception response to any link of a data chain; except the final link. Even though an exception response is sent for a link of a chain, CICS/VS may have already sent the rest of the chain; therefore, the subsystem must to purge the rest of the chain. When a cancel indicator is received,the subsystem should ignore the data chain currently being received and leave the in-chain processing state.

LOGICAL UNIT I/O ERROR HANDLING

The node abnormal condition program (DFHZNAC) is a system program responsible for processing all abnormal situations associated with a logical unit. This is analagous to the situation under BTAM support in which terminal abnormal condition program (TACP) is scheduled to resolve terminal errors. For VTAM-supported logical units, however, all information concerning the processing state of the terminal is contained in the TCTTE and RPL. No accompanying line entry exists for a logical unit as is the case for a BTAM-supported terminal. Consequently, when a terminal error must be handled for a logical unit, the TCTTE itself is placed onto the system error queue.

The detection of abnormal conditions associated with logical unit operations causes CICS/VS to schedule DFHZNAC. DFHZNAC is scheduled for a TCTTE any time that an operation requested by CICS/VS from VTAM completes in error, or is rejected and cannot be performed. The receipt of an exception response sent by a logical unit also causes DFHZNAC to be scheduled to permit analysis of the sense information and issuance of any appropriate messages.

When DFHZNAC is scheduled, it analyzes the situation and determines the appropriate action to take. Before the action is taken, NEP is scheduled to determine whether the user agrees with the proposed solution. To assist NEP, DFHZNAC sets flags, indicating the proposed action. These action flags are in DFHZNAC's transaction work area (TWA). In most cases, NEP can modify DFHZNAC's proposed actions. The only time that DFHZNAC overrides NEP's modification of the TWA is when a terminal is to be disconnected from CICS/VS; that is, when DFHZNAC determines that the abnormal situation requires that CICS/VS issues the VTAM CLSDST macro instruction for a logical unit. In such a case, the eventual action will depend on the system sense code received. When control is returned to DFHZNAC from NEP, DFHZNAC performs the actions specified in TWAOPTL (except when disconnecting terminals, as noted above), issuing messages and setting error codes, as necessary.

The system programmer needs to code a node error program (NEP) only if he wishes to perform additional error processing beyond that performed by DFHZNAC. DFHZNAC gives control to NEP by issuing a DFHPC TYPE=LINK macro instruction. DFHZNAC also passes the address of the TCTTE concerned, so that the system programmer can specify further recovery actions based on the processing state of the logical unit. When NEP has performed its functions, it returns control to DFHZNAC by issuing a DFHPC TYPE=RETURN macro instruction.

Upon entry to NEP, the following fields are available to the system programmer:

- The error code generated by DFHZNAC.
- The action flags set by DFHZNAC.
- The address of the TCTTE.
- The terminal name.
- The sense codes received by DFHZNAC:

Symbolic labels for error codes and action flags are provided in the NEP coding released with CICS/VS. Linkage to NEP is provided by CICS/VS.

If DFHZNAC is scheduled because of the receipt of an exception response or a VTAM LU status indicator, the sense information in the TCTTE is available to DFHZNAC and NEP to determine any necessary actions. If DFHZNAC is scheduled because of loss of the connection between CICS/VS and a logical unit, DFHZNAC abnormally terminates any transaction in progress at the time of the failure. NEP analysis and processing is permitted, but message retry should not be attempted.

The DFHZNAC error message is sent to the master-terminal log prior to linking to NEP. User-written messages may also be sent to the log using the transient data facility. To write the installation's own messages, the system programmer must code the DFHTD TYPE=PUT macro instruction directly into NEP.

USER EXIT-ROUTINES FOR CICS/VS DFHZCP

CICS/VS advanced communication support provides the system programmer with the option of coding a user exit-routine which is to be given control at defined points during the processing of a request by DFHZCP.

Control is given to the specified exit-routine at each of the following three points every time a request referring to a VTAM-supported TCTTE is serviced:

- Before a task attach.
- Before issuing the logical message in the DFHZCP send subroutine; no chaining requirements have yet been determined.
- After the entire logical message is received by CICS/VS.

Part 2. The Components of CICS/VS

This part deals with the components and functions that make up CICS/VS. There are six chapters, corresponding to the six components:

Chapter 6. System Management

TASK MANAGEMENT

The ability to process more than one transaction concurrently is provided by Task Management. Tasks are scheduled and processed by priority, control being given to the dispatchable task with the highest priority. Task priority is the sum of the priorities assigned to the transaction code, the terminal, and the terminal operator. The number of active tasks is limited by the amount of available address space and the number of tasks specified as a maximum for the task class by the user.

Task Management (DFHKCP) can be divided into support for the DFHKC macro and the Task Dispatcher.

DFHKC MACRO SUPPORT

Issuing a DFHKC macro instruction causes a code to be set in the requesting task's Task Control Area (TCA), and a call to be made to Task Management.

Except during task termination, the requesting task's TCA is used by Task Management to communicate with Storage Management. Control is not always returned directly to the requesting program, but may pass to the Task Dispatcher which selects the task to be given control.

Initiate a Task (ATTACH)

Task Management validates transactions by checking the Program Control Table (PCT), which lists all the valid transaction codes and their associated programs, so that control may be transferred to the correct program.

If a match is found in the PCT then operator security, message protection, and task class maximum are verified and storage obtained for the new TCA and a Dispatch Control Area (DCA). The dispatch priority is set in the DCA. Control returns to the caller for conditional requests, and to the Task Dispatcher for unconditional requests. If the transaction is not valid then the abnormal condition program is attached.

On starting up a task, Task Management acquires a TCA through Storage Management (described later in this chapter). If necessary, the TCA may also include a Transaction Work Area (TWA), which may be used by an application program during the life of a transaction. If a task uses anticipatory paging then the initial area obtained contains the TCA, TWA (if any), and space for data areas. The size of these areas is obtained from the PCT. The TCA and TWA are released when the task terminates.

For CICS/OS/VS, if the task is marked in the PCT as a primed task, a Primed Allocation (PRA) is obtained. This area will contain the TCA, and will also be used for rapid storage allocation for some areas used by the task.

Terminate a Task (DETACH)

The task is disconnected from the associated terminal, and any deferred work is processed. Any interval control elements for the task are cancelled, terminal management is notified, storage for the task is freed, and the routine exits to the task dispatcher. The call to Storage Management to free the TCA will also result in storage still chained to the TCA being freed.

Enqueue upon a Resource (ENQ)

If the resource is unknown, the ENQ routine creates a Queue Element Area (QEA) and returns to the caller. If the resource is known and already held by the caller, the ENQ routine simply increments the use count. If the resource is known and not held by the caller, then the caller's DCA in placed in a wait chain and the calling task is suspended.

Dequeue upon a Resource (DEQ)

The resources held by the calling task are checked against the resource being dequeued. If the resource is found, its use count is decremented. If the count reaches zero, the QEA is moved from this task chain and either given to a waiting task (which is started up) or, if there is no such task, freed.

Dequeue all Resources (DEQALL)

The resources held by the calling task are all released, and their QEAs freed or transferred to tasks requiring the resources.

Change Priority of a Task (CHAP)

The DCA for the task is removed from the active dispatching chain and immediately replaced on the chain so that it has the required priority. Control passes to the Task Dispatcher.

Synchronize a Task (WAIT)

There are several forms of WAIT. WAIT dispatchable just allows a higher priority task to be dispatched. A WAIT with DCI=CICS means that the posting is done within CICS/VS and so the ECB is never added to the operating system wait list when CICS/VS issues a system WAIT. A Terminal Management WAIT falls through into a Task Management WAIT which causes the task to be moved to the SUSPEND chain. WAITs for one or more ECBs leave the TCAs on the active chain.

Suspend a Task (SUSPEND)

The task DCA is moved from the active task chain to the suspended task chain, if necessary in timeout sequence. Tasks are suspended because they are likely to be waiting for a relatively long time, that is when waiting for a resource or for terminal I/O, but not for disk I/O. CICS/VS system tasks may not be suspended.

Resume a Task (RESUME)

The task DCA is moved from the suspended task chain to the active task chain, positioned by priority. RESUME is issued by Terminal Management

when terminal I/O completes, and by other modules when resources become available.

Schedule a Resource (SCHEDULE)

SCHEDULE is used by Time Management, Transient Data Management, BMS and Asynchronous Task Processing to tell Task Management that a task should be started. Task Management cannot actually start the task until the required terminal is available. It therefore records all the necessary information in an Automatic Initiate Descriptor until the terminal is free, that is, when Terminal Management has issued an AVAIL macros.

Declare Resource Availability (AVAIL)

This macro is issued by Terminal Management to indicate that a terminal is now free. The Automatic Initiate Descriptor chain is searched for a match with the Terminal Control Table Terminal Entry (TCTTE). A task is attached and control returns to the caller.

HPO Services

Services have been added to Task Management for OS/VS2 versions of CICS/VS for the High Performance Option. These are:

- Switch to SRB mode
- Switch to TCB Mode
- Attach HTA
- Detach HTA

TASK DISPATCHER

The Task Dispatcher maintains an active task chain and a suspended task chain. The Dispatcher scans the active task chain to find a dispatchable task. The suspended task chain is scanned only during stall processing. The elements on the chains are Dispatch Control Areas (DCAs), each DCA containing pointers to the DCAs of next higher and lower priority, and to the TCA of the associated task. The CSA contains the addresses of the highest and lowest priority DCAs on each of the chains. The dispatchability of a task is determined by examining the setting of the dispatch control indicator in the DCA. If it indicates that the task is waiting on the completion of some event, the control block associated with the event is tested for completion posting. The suspended task chain is a FIFO chain, without regard for task priority. Before looking at the active task chain, the Dispatcher examines the first timeout task on the suspended task chain. If the timeout interval has expired then the task is purged.

If the Dispatcher determines that a task is dispatchable, its TCA is activated, the task's register contents are restored, and control passes to the program which was in control of the TCA when the wait state was entered. If the Dispatcher finds no CICS/VS task that is dispatchable,it releases control of the CPU to the operating system,requesting that control be returned to CICS/VS upon completion of the next event. Before doing so, an operating system interval timer macro is issued so that CICS/VS can always get control back.

The Dispatcher uses its own TCA to communicate between modules. It branches into Time Management where time-ordered events are initiated as request times expire. The time remaining until expiration of the next time-ordered event is returned to the Dispatcher, and is used when operating system timer services are requested.

The Dispatcher enforces task class maxima. It also detects and corrects some error conditions. It will purge transactions in order to overcome the system stall condition, and will terminate tasks which are identified as exceeding their runaway task or timeout limits.

The System Stall condition occurs when main storage resources available to CICS/VS become overloaded to the point where active transactions cannot continue processing and new transactions cannot be initiated. Corrective action involves the purging of low-priority purgeable transactions so designated by the user.

The Runaway Task condition occurs when an application program may have developed an endless loop within the program logic. Corrective action involves the abnormal termination of the task.

The Read Time-Out condition occurs when a transaction waits for a terminal input message beyond the time specified. Corrective action includes the abnormal termination of the task.

At each invocation the Task Dispatcher refreshes the current time-of-day values in the CSA, and checks if the time requires the Terminal Management task to be dispatched.

In OS/VS2 CICS/VS systems, if the Terminal Management task is not dispatched, Task Management will check the Service Request Block (SRB) chain and dispatch the first SRB found, without examining the DCA chains. Also the maximum task-in-class count is replaced by an active task-in-class count.

STORAGE MANAGEMENT

All dynamic storage for CICS/VS and for the user-written application programs is controlled by Storage Management (DFHSCP). Requests to acquire or release dynamic storage are communicated to Storage Management by CICS/VS macro instructions. CICS/VS management functions use dynamic storage for input/output areas, program load areas, and user-defined transaction work areas. User-written application programs use dynamic storage for intermediate work areas and for transaction processing.

An optional user exit is provided upon initial entry to Storage Management.

Storage Initialization

Each byte of dynamic storage may be initialized to a bit configuration specified as an option in the requesting macro instruction. For example, a dynamic storage area can be initialized to binary zeros or to EBCDIC blanks.

Storage Accounting

All storage areas used by a transaction are chained. This allows CICS/VS to release all dynamic storage associated with a transaction either upon request by the user or when the transaction is terminated.

Dynamic Storage Verification and Reclamation

CICS/VS verifies all requests for dynamic storage. This helps in isolating and terminating transactions that inadvertently destroy storage chains. If storage chains are destroyed, CICS/VS attempts to reconstruct them so that dynamic storage remains usable by CICS/VS applications.

Conditional Storage Acquisition

The user can issue either a conditional or unconditional request to acquire dynamic storage for a transaction. When there is insufficient dynamic storage to satisfy a conditional request, control is returned to the user with an indication that there was insufficient dynamic storage; an unconditional request results in the transaction being suspended until sufficient storage is available, and also prevents any new tasks being initiated until storage is available. In this way, Storage Management moderates the effects of high dynamic storage demands and keeps CICS/VS running.

System Overload Detection

To relieve an overload, CICS/VS uses a technique that involves a storage cushion. The cushion is a quantity of dynamic storage held in reserve by CICS/VS. When a request for dynamic storage cannot be satisfied, the cushion is released for use by current tasks and initiation of new tasks is inhibited. When the demands for dynamic storage have diminished, the cushion is reacquired by CICS/VS and new transactions can be initiated.

Storage Statistics

Statistics maintained by Storage Management include the number of requests for storage acquisition, the number of times processing must be delayed for storage acquisition, and the number of requests for storage disposition.

Storage Control Services

The following services are provided automatically by Storage Management without communication from the application program.

Storage Request Enqueuing and Dequeuing:

Enqueues and dequeues requests for dynamic storage that cannot be satisfied immediately.

System Notification:

Inhibits the initiation of new work. This is an internal function; no external notification is given.

Storage Accounting:

Automatically chains storage acquired by a transaction so that a single macro from Task Management is sufficient to free any remaining storage upon termination of the transaction.

The following services are performed by Storage Management in response to specific requests from either an application program or another CICS/VS function.

Storage Acquisition:

Allocates dynamic storage fo_ a transaction.

Storage Disposition:

Releases dynamic storage for a transaction.

Storage Initialization:

Initializes allocated dynamic storage to the bit configuration specified by the user in the macro instruction.

THE STORAGE MANAGEMENT MODULE

Storage Management (DFHSCP) communicates with other CICS/VS functions and user-written application programs to satisfy their storage requirements. It makes extensive use of CICS/VS control blocks and interfaces with other CICS/VS functions for special processing when exception conditions arise.

All CICS/VS and user-written programs communicate their requests for Storage Management services through the TCA. The address of the storage acquired is returned to the requesting program in the TCA.

Storage Management determines the type of request by referring to the TCA and returns the address of acquired storage in the TCA. Storage associated with a task is chained off the TCA.

When working with terminal storage, the address of the TCTTE is found in the TCA. Storage Management maintains a chain of terminal storage through a field in the TCTTE. A count of storage violations associated with a terminal is also kept in the TCTTE.

Storage Management manipulates the short on storage (SOS) indicator in the CSA. The CSA is used to find the Page Allocation Map (PAM) and the suspended Dispatch Control Area (DCA) chain. Statistics related to storage control are kept in the CSA.

Whenever there are tasks suspended for lack of storage, some storage is released, Storage Management searches the chain of DCAs and attempts to fulfil a suspended request for allocation.

When CICS/VS is running out of storage, Storage Management searches the Processing Program Table (PPT) for programs residing in dynamic storage that are not currently in use and are not permanently resident. Storage for such programs is freed, and the PPT is changed to reflect the fact that the program is no longer in storage.

A count of the number of storage violations associated with a transaction identification is kept in the Program Control Table (PCT).

The Page Allocation Map contains dynamic values relating to the allocating and freeing of storage. It also contains a map of all pages

in the dynamic storage area that indicates the current disposition of each page.

If an unconditional request for storage cannot be satisfied, Storage Management issues a DFHKC TYPE=SUSPEND macro instruction, which is a Task Management macro instruction used only by CICS/VS management modules. The requesting task is suspended until the storage request can be satisfied, at which time a DFHKC TYPE=RESUME macro instruction (also used only by CICS/VS management modules) is issued to request Task Management to start the task.

If an invalid request is issued, or an address specified in a request is invalid, Storage Management returns control to the calling program, setting the INVREQ response code.

If Storage Management has a program check while attempting to service a DFHSC TYPE=GETMAIN or DFHSC TYPE=FREEMAIN request, the System Recovery program intercepts and passes control to the Storage Control Recovery routine.

If Storage Management detects a storage violation while servicing a DFHSC TYPE=GETMAIN or DFHSC TYPE=FREEMAIN request, control is passed to the Storage Control Recovery routine. The Storage Control Recovery routine tries to recover from the storage problem and, if successful, returns to Storage Management so that the request that revealed the problem can be retried, otherwise CICS/VS is terminated.

For a primed task in CICS/OS/VS, Storage Management will allocate some areas from the Primed Allocation (PRA) obtained with the TCA. Storage Management does not free areas in the PRA until task termination.

PROGRAM MANAGEMENT

Program Management controls and supervises CICS/VS application programs. Program Management macro instructions are used to load programs, link or transfer control to programs, delete a loaded program, abnormally terminate a task, and return control from a program. The normal termination of tasks and the termination of transaction processing are additional services initiated by this program. Single copies of application programs in dynamic storage are controlled to allow concurrent use by multiple tasks.

When a requested application program is already in dynamic storage, Program Management transfers control directly to that program. Each program's location on a direct access volume and in main storage, if applicable, is kept in the Processing Program Table (PPT). The status of each program is maintained in the PPT.

Programs are stored in a CICS/VS library in relocatable format and are accessed through the CICS/VS asynchronous program fetch data facilities. This loading facility allows CICS/VS execution to continue during program load time. For CICS/DOS/VS, the real-time relocatable program library is a standard core-image library prepared by the DOS/VS linkage editor. For CICS/OS/VS, the real-time relocatable program library is a standard partitioned data set. Programs are prepared for this library by the OS/VS linkage editor. For CICS/OS/VS, users can concatenate other private libraries to this library.

As control for a task is passed from one processing program to another and returned, Program Management saves and restores the general purpose registers. When control is returned to Program Management at

the completion of task processing, it issues a request to detach the task which, in turn, releases all dynamic storage associated with the task.

Application programs remain resident in dynamic storage when not being used, unless there is an indication that system storage resources have become overloaded. In this case, programs not currently in use are deleted from dynamic storage.

An optional user exit is provided from Program Management after program fetch.

PROGRAM MANAGEMENT SERVICES

The following functions are automatically performed by Program Management without communication with the application program.

High Level Language (HLL) Macro Interface:

Intercepts HLL macro-level (not command-level) requests for CICS/VS services to save information, passes control to the appropriate CICS/VS management function, and subsequently returns to the HLL program.

Program Purge

Causes unused application programs to be purged from dynamic storage when CICS/VS is in the overload state. (Once used, an application program remains resident in dynamic storage, even when not in use.)

Asynchronous Program Fetch

Provides for the fetching of application programs from the direct access storage device into dynamic storage while allowing other transaction activity to proceed during the I/O operation.

The following services are performed by Program Management in response to a specific request from either a user application program or another CICS/VS function.

Link

Retains requesting program and its general registers for subsequent return and passes control to the program specified in the macro instruction. The linked-to program is considered to be at the next lower logical level.

Transfer Control

Transfers control from one application program to another without return being possible. The target program is considered to be at the same logical level as the calling program.

Load

Loads the designated program into dynamic storage and returns its entry address to the requesting program.

Delete

Releases the designated program, which was previously loaded.

Return

Returns control to the program, possibly CICS/VS, considered to be at the next higher logical level. If the task is at the highest level, this leads to normal transaction termination.

Abend

Terminates a transaction and its related task and passes control to the abnormal condition program.

BLDL

Supports the CSMT NEWCOPY function by obtaining the information about the new program.

THE PROGRAM MANAGEMENT MODULE (DFHPCP)

Program Management (DFHPCP) may communicate with any CICS/VS function or any user-written application program. In addition, CICS/VS-supported high-level language programs interface with CICS/VS through Program Management. ANS COBOL and PL/I modules, using the macro-level interface, establish fields in the TCA and communicate with Program Management when invoking other CICS/VS functions.

Any CICS/VS or user-written module may issue a Program Management DFHPC TYPE=ABEND macro instruction to request that a task be abnormally terminated.

TIME MANAGEMENT

The following services are performed by CICS/VS, based on intervals of time specified by the user during system initialization. They require no communication with user-written application programs.

CICS/VS Exit Time Interval Control

The CICS/VS exit time interval is the maximum interval of time for which CICS/VS wishes to release control to the operating system in the event there are no transactions ready to resume processing.

System Stall Detection and Correction

Provides automatic detection of the system stall condition when the CICS/VS dynamic storage resource becomes overloaded to the point where no active transactions can continue and no new transactions can be

initiated. Corrective action involves the purging of low-priority transactions designated as purgeable by the user. The status of each transaction is defined and controlled by the user.

Runaway Task Detection and Correction

Automatic detection of the situation where an application program may have developed a loop within the program logic. Corrective action involves the abnormal termination of the transaction.

The following services are performed by Time Management in response to a specific request from either an application program or another CICS/VS function.

Time of Day

Provides the ability to retrieve the current time of day in either binary or packed decimal format.

Time-Dependent Transaction Synchronization

Provides the user with three optional services:

- WAIT, permits a transaction to temporarily suspend itself for a given period of time. When the time has elapsed, the transaction resumes execution.
- POST, provides the means for a transaction to be notified when the specified interval of time has elapsed or the specified time of day occurs. The transaction proceeds to execute while the time interval is elapsing.
- CANCEL, allows a transaction to terminate its own or another transaction's request for a WAIT or POST service.

Automatic Time-Ordered Transaction Initiation

Provides for the automatic initiation of a transaction at a specified time of day (or after a specified interval of time has elapsed) and for the control of data that is to be accessed by the transaction. The user can also cancel a pending request for automatic time-ordered transaction initiation.

Optional user exits are provided as follows:

- Before determining what type of request for time services was issued.
- Upon expiration of a previously requested time-dependent event.

THE TIME MANAGEMENT MODULE (DFHICP)

Time Management can be divided into two functional areas. The first of these services Interval Control (DFHIC) macro instructions and is executed under control of the requesting task's TCA. The second detects

the expiration of time-dependent events, and is performed by the Task Dispatcher.

Issuing a DFHIC macro instruction causes a request code to be set in a field in the requesting task's TCA. Processing enters the Time Management macro instruction service logic. The register contents are saved in the TCA and the requested service is performed or initiated. While performing the service, Time Management interfaces with other CICS/VS management functions (Storage Management, Task Management, and Temporary Storage Management). Time Management builds Interval Control Elements (ICEs) for time-dependent requests made using DFHIC macro instructions. The ICEs are chained off the CSA in expiration-time sequence. Control is returned to the requesting CICS/VS module or user-written program when the requested service has been queued or performed.

The Task Dispatcher activates Task Management's TCA for intermodule communication and branches to the expiration analysis logic of Time Management program. This logic checks the ICE chain entries, in expiration time sequence, determining whether or not each has expired. Each expired ICE is removed from the chain, and processing related to the service requested through the DFHIC macro instruction is performed or initiated. During expiration analysis, Time Management interfaces with Storage Management and uses Task Management services as well. Since the normal processing flow in the Task Management program is from the requested service logic, the expiration analysis logic is reentered by again activating Task Management's TCA.

One of the functions of the Task Dispatcher is to refresh the current time-of-day values retained in the CSA. During this process, the clock will be reset if midnight is detected.

During expiration analysis, a CICS/VS system task can be started which will adjust the expiration times of day to reflect the occurrence of midnight and to reset the CICS/VS current time-of-day and date values to coincide with the operating system. The CICS/VS-provided Time Adjustment program performs these functions by adjusting the expiration times in the ICEs (as well as other CICS/VS-maintained expiration times), and then changing the time-of-day values in the CSA to equal the operating system clock.

TERMINAL MANAGEMENT

Terminal Management provides communication between terminals and user-written application programs. The user can specify that concurrent terminal support can be provided by any combination of the following access methods:

- Basic telecommunication access method (BTAM)
- Virtual telecommunication access method (VTAM), with the network control program (NCP)
- Graphics access method (GAM) (CICS/OS/VS only)
- Telecommunications access method (TCAM) (CICS/OS/VS only)

Terminal Management uses data that describes the communication lines and terminals. This data is kept in the Terminal Control Table (TCT), which is generated by the user as part of the environment definition. The table entries contain terminal request indicators, status, statistics, identification, and addresses of I/O and related areas. The

primary function of Terminal Management is to take an I/O request and convert it to a format acceptable to the access method.

A terminal connected to BTAM is polled to request initial input. The user specifies the terminal device characteristics and desired polling interval so that CICS/VS is generated to satisfy his requirements. (A polling interval is the period of time between one attempt to read data from a terminal or group of terminals and the next attempt to read data from the same terminal or terminals.)

For BTAM devices, when a read is completed the input data is converted to the extended binary coded decimal interchange code (EBCDIC). When there is need for a task to process a message, a CICS/VS Task Management ATTACH macro instruction is issued by Terminal Management. When data is to be transmitted to a terminal, processing programs execute a CICS/VS Terminal Management WRITE macro instruction. The translation of output data from EBCDIC to the appropriate terminal code is performed if required.

VTAM network functions allow terminals to be connected to any control subsystem that is online. This enables a terminal operator to switch from one CICS/VS system to another or to another subsystem, such as IMS/VS.

The functions of Terminal Management are categorized as either transmission facility control functions (those functions that are normally related to the control of the communication lines) or terminal device-dependent control functions (those terminal functions that are dependent upon device type and access method).

When TCAM controls communication lines under CICS/VS/OS, those lines are no longer dedicated to the CICS/VS region. Thus the a single terminal can access programs in separate regions supported by TCAM. TCAM facilities available within the region supported by TCAM include message switching, broadcasting, disk queuing, checkpoint/restart of the communication network, and TCAM terminal support.

TESTING FACILITY

To allow the user to test programs, the sequential access method (SAM) is used to control sequential devices such as card readers and printers, magnetic tape, and direct access storage devices. These sequential devices can be used to supply input/output to CICS/VS before actual terminals are available or during testing of new applications.

TERMINAL MANAGEMENT SERVICES

The following services are performed by, or in conjunction with, Terminal Management.

- Service request facilities
- System control services
- Transmission facilities
- BTAM device-dependent services

Service Request Facilities

Write Request:

Sets up and issues or queues access method macros; performs journaling and journal synchronization.

Read Request:

Sets up and issues access method macro instruction; performs journaling if required.

Wait Request:

Issues CICS/VS WAIT.

Dispatch Analysis:

Determines the type of access method and terminal used, and executes the appropriate area of Terminal Management.

System Control Services

Automatic Task Initiation:

Services requests for automatic task (transaction) initiation caused by events internal to the processing of CICS/VS.

Task Initiation:

Requests the initiation of a task to process a transaction from a terminal. When an initial input message is accepted, a task is created to do the processing.

Terminal Storage:

Performs allocation and deallocation of terminal storage if requested by application program.

Transmission Facilities - VTAM

Connection Services:

Accepts logon requests, requests connection of terminals for automatic task initiation, and returns terminals to VTAM, as specified by the user.

Transmission Facilities - BTAM

Translation:

Translates data received from transmission code to EBCDIC and data to be sent from EBCDIC to the appropriate transmission code.

Line Advance:

Scans the terminal control table to make line control information available for analysis.

Line Analysis:

Analyzes the terminal control table line control information to determine which terminal facilities require further action. For example, an indication that a communication line is free may indicate that a polling operation should begin.

Event Termination:

Provides a reset poll for certain terminal devices to service write requests.

Transmission Facilities - BTAM/VTAM

Access Method Selection:

Passes control to the appropriate access method routine based on the access method specified in the terminal control table.

Wait:

Synchronizes the terminal control task with all other tasks in the system. When all terminal read and write operations are begun that it ispossible to initiate, Terminal Management processing is complete and control is returned to Task Management to allow dispatching of other tasks.

Transmission Facilities - TCAM

Message Control Program Facilities:

Invites and selects terminals to transmit or receive data, manages dynamic buffers, handles messages and directs the flow of data through the system on a priority basis, maintains queues in main storage and on direct access devices for terminals and application programs, and handles error checking, operator control, and checkpoint/restart.

BTAM Device-Dependent Services

Input Event Treatment:

Processes a completed input event, including error checking, storage management, translation, and task initiation.

Output Event Treatment:

Processes a completed output event, including error checking and storage management.

Activity Control:

Examines the control information for each terminal, checking for requested WRITE, READ, WAIT, and other Terminal Management macro instruction requests.

Input Event Preparation:

Prepares the line for an input event, including Storage Management.

Output Event Preparation:

Prepares the line for an output event, including translation.

Event Initiation:

Prepares the terminal data event control block and the linkage of terminal device-dependent control to the appropriate access method.

TERMINAL ERROR RECOVERY

The resolution of permanent transmission errors involves both CICS/VS and additional user coding. CICS/VS cannot arbitrarily take all action with regard to these errors. User application logic will sometimes be necessary to resolve the problem. For the portion of the telecommunications network connected to BTAM, TCAM, or GAM, these services are provided by the terminal abnormal condition program (TACP) and by the user-written, or sample, terminal error program (TEP). For the VTAM part of the network, terminal error handling is carried out by the node abnormal condition program (NACP), the sample node error program (NEP), provided by CICS/VS, or a user-written node error program.

The following sequence of events takes place when a permanent error occurs for a terminal:

1. The terminal is placed in an out of service status.
2. The terminal/node abnormal condition program is attached to the system to run as a separate CICS/VS task.
3. The terminal/node abnormal condition program writes the error data to a destination in transient data control if the user has defined one. This destination is defined by the user and may be intrapartition or extrapartition.
4. The terminal/node abnormal condition program then links to the appropriate terminal/node error program to allow terminal/transaction-oriented analysis of the error. In the terminal/node error program, the user may decide to have the terminal placed in service, have the line placed in or out of service, or have the transaction in process on the terminal abnormally terminated.
5. The terminal/node abnormal condition program is detached from the system.

THE TERMINAL MANAGEMENT MODULES (DFHTCP, DFHZCP)

Terminal Management consists of two CICS/VS modules, DFHZCP and DFHTCP. DFHZCP provides both the common (VTAM and non-VTAM) interface as well as the VTAM-only support. DFHTCP provides the non-VTAM support. Terminal Management communicates with application programs, CICS/VS System Management functions (Task Management, Storage Management) CICS/VS Application Services (Basic Mapping Support and Data Interchange Program), System Reliability functions (abnormal condition handling), and with operating system access methods (BTAM, GAM, SAM, VTAM, or TCAM). Requests for Terminal Management functions made by application programs, BMS, or Task Management, are processed through the common interface of DFHZCP. Generally, Terminal Management requests for other CICS/VS or operating system functions are issued by either DFHZCP (VTAM support) or DFHTCP, depending upon the terminal being serviced.

DFHTCP and DFHZCP are two separate modules. They are always assembled separately and loaded separately. DFHZCP is always generated because it contains some internal routines which are necessary for the successful operation of DFHTCP. VTAM support within DFHZCP is generated by specifying ACCMETH=VTAM in the DFHSG PROGRAM=TCP macro instruction. The ACCMETH and VTAMDEV operands must be coded to support a CICS/VS advanced communication system.

Common Interface

When a Terminal Management (DFHTC) macro instruction is issued by an application program, by the Data Interchange Program, or by Basic Mapping Support (BMS), request bits are set in the user's TCA and control is passed to the common interface (VTAM,non-VTAM) routines of DFHZCP.

If the DFHTC macro instruction includes a WAIT request, control is passed to Task Management to place the requesting program (task) in a suspended state. If a WAIT request is not included, control is returned to the requesting task.

A field in the task's TCA contains the Facility Control Area Associated Address, a pointer to the terminal with which the task is associated.

Task Management dispatches Terminal Management through the common interface for one of the following reasons:

- The system partition/region exit time interval (specified in the ICV operand of the DFHSIT macro instruction by which the System Initialization Table is generated) has elapsed.
- The Terminal Management event initiated by the DFHTC macro instruction has been posted complete (non-VTAM ECB posted or exit scheduled in the case of VTAM).
- Neither of the previous events, above, has occurred, but one second has elapsed since the last time Terminal Management was dispatched with a pending request to be serviced.

Terminal Management, through its common interface, requests Task Management to perform a CICS/VS WAIT when Terminal Management has no further work that it can do.

Access Method Dependent Interface

Terminal Management communicates with Storage Management using a DFHSC TYPE= GETMAIN or FREEMAIN macro instructions to obtain and release storage as follows:

Non-VTAM

DFHTCP issues DFHSC macro instructions to obtain and release terminal and line storage.

VTAM

DFHZCP issues DFHSC macro instructions to obtain and release terminal, line (line class is used for the Receive-Any I/O areas), and RPL storage.

Terminal Management communicates with Task Management using a DFHKC macro instruction. The macro instruction is issued by DFHZCP or DFHTCP, depending upon the terminal being serviced. Terminal Management may request Task Management to perform one of the following:

- Attach a task upon receipt of a transaction identification from a terminal.
- Respond to a DFHKC TYPE=WAIT request.
- Respond to a DFHKC TYPE=RESUME request.
- Respond to a DFHKC TYPE=AVAIL request (a Task Management macro instruction documented only for system programming) when a time-initiated task is indicated for a terminal and that facility is available.

Terminal Management communicates with operating system access methods in either of the following ways, depending upon the terminal being serviced:

Non-VTAM (DFHTCP)

DFHTCP builds access method requests in the DECB, which is part of the TCTLE. The DECB portion of the TCTLE is passed to the access method by Terminal Management to request a service of that access method. The access method notifies Terminal Management of the completion of the service through the DECB. Terminal Management analyzes the contents of the DECB upon completion to determine the type of completion and to check for error information.

VTAM (DFHZCP)

DFHZCP builds VTAM request information in the RPL, which is then passed to VTAM for servicing. VTAM notifies Terminal Management of completion by placing completion information in the RPL. DFHZCP analyzes the contents of the RPL upon completion to determine the type of completion and the presence of error information. Communication with VTAM also occurs by VTAM scheduling exits. VTAM passes parameter lists and does not always use the RPL.

Terminal Management communicates with the CICS/VS abnormal condition functions in either of the following ways, depending upon the terminal being serviced:

Non-VTAM

DFHTCP attaches the Terminal Abnormal Condition program (TACP) and passes a Terminal Abnormal Condition line entry (TACLE) when an error occurs. The TACLE is a copy of the DECB portion of the TCTLE and contains all information necessary for proper evaluation of the error, plus special action indicators that can be manipulated to alter the error correction procedure. After the DECB has been analyzed, it is passed to the user's error recovery program (DFHTEP).

VTAM

DFHZCP attaches the Node Abnormal Condition Program (NACP) when an error occurs, again passing information and special action indicators. DFHZNAC does some preliminary error processing and then passes control to the user's Node Error Program (DFHNEP). Upon the completion of the user's error processing, control is returned to DFHZNAC.

Terminal Management executes either under the user's TCA or its own TCA as follows:

User's TCA

- During the application program interface
- During the interface with Basic Mapping Support
- While performing non-chained VTAM terminal requests

Terminal Control's TCA

- When Task Management dispatches Terminal Management
- When Terminal Management issues a request to Task Management
- When Terminal Management issues a request to Storage Management
- While performing non-VTAM terminal I/O or chained VTAM terminal I/O

High Performance Option

For CICS/VS on OS/VS2 VTAM will be used with CICS/VS as an authorized program so that the VTAM path length is reduced. This is achieved by dispatching SRBs to actually issue the reads and writes to the terminals. All the SRB code is executed in the module DFHZHPRX.

FILE MANAGEMENT

Access to the user data base is provided by File Management, which consists of the module DFHFCP (and, in OS, the module DFHFCD), the File Control Table (FCT), and access method dependent logic for each access

method described to the system. File Management reads from and writes to user-defined data sets, gathers statistics, and acquires dynamic storage for data base operations. File Management uses control information defined by the user in the FCT. This table describes the physical characteristics of all the data base data sets and any logical relationships that may exist between them.

The following access methods are used by File Management, at the option of the user:

- VSAM virtual storage access method
- ISAM indexed sequential access method
- BDAM or DAM direct access method for OS/VS or DOS/VS,

File Management provides the following services and features:

- Random record retrieval
- Random record update
- Random record addition
- Random record deletion (VSAM only)
- Sequential record retrieval
- Segmented records
- Indirect accessing
- BDAM (or DAM) deblocking
- Logical open/close of data sets
- Exclusive control of records during update operations
- DOS/VS ISAM variable-length records
- LOCATE mode, read-only retrieval (VSAM only)
- Mass record insertion (VSAM only)
- Automatic journaling

SEGMENTED RECORDS

A segmented record is one in which the components of the record have been identified and grouped according to frequency of use, function, and logical relationship. The identifiable groups are called segments. Some segments, such as those that contain identification or major record control fields, appear in all records. Other segments may appear only in certain records. With the segmented record capability, it is possible to retrieve an individual record or selected segments of an individual record. The primary reason for segmenting records is to conserve dynamic storage and, in the case of variable-length records, to conserve direct access storage.

The user defines the record segments to CICS/VS in the FCT. In addition, the first segment of each record to be retrieved in segments

must contain control information to indicate the presence or absence of each segment. A segment should contain logically related data so that only selected segments are required to satisfy the processing requirements of a transaction. A transaction that uses only selected record segments requires less dynamic storage for its processing than it would if all record segments were read into the storage area.

The user, in selecting those segments necessary for processing transaction groups, identifies them to CICS/VS as a part of the data set definitions in the FCT. Such a group of segments is a segment set. A segment set can include a single segment or all segments of a record. When a request identifying a segment set is made to file control, CICS/VS always returns the header control segment plus the segments in that set.

Segmented records can be used with either DAM or ISAM data set organizations but are especially suited for VSAM data sets.

DEBLOCKING SERVICES FOR DAM DATA SETS

CICS/VS provides deblocking of logical records on a direct access (BDAM or DAM) data set. This service is provided for both fixed- and variable-length records. The data set must have been created according to standard System/370 record formatting conventions.

INDEX DATA SETS INDIRECT ACCESSING

CICS/VS (optionally) allows the use of cross-index data sets to access another data set, which may be the main data set or another level of index data set. If a record retrieved from a cross-index data set indicates multiple entries in the main data set, information is returned to the user-written application programs to be used in selecting the appropriate main data set entry. When the cross-index does not indicate multiple entries in the main data set, File Management reads the requested record from the main data set.

Organization of the cross-index data set may be either VSAM, ISAM, or BDAM (DAM). The index record contains, in addition to the information used to find it, the search argument for the record on the data set that the index data set references. The index record can contain any other information desired by the user. The location of the search argument, its length, and the data set identification for the referenced data set are supplied to CICS/VS as part of the data set definition in the FCT.

DOS/VS ISAM VARIABLE-LENGTH RECORDS

CICS/DOS/VS supports the retrieval and static update of variable-length records within fixed-length blocks under an ISAM organization. These pseudo-variable blocks must conform to System/370 variable-length record format conventions. That is, the first four bytes must contain the block length in the form LLbb. Since all blocks are fixed length, this value is the same for all blocks. Each logical record within the block must reflect the length of the record in the first four bytes (LLbb). A logical record cannot be continued onto the next block. The first byte of any unused portion of a block must contain a hexadecimal FF.

The addition and deletion of records on a DOS/VS ISAM variable-length record data set must be handled by the user in offline batch processing. When creating the data set, it must be defined as fixed unblocked, and the key for each block must be the same as the last logical record in that block. The block size must be an even number of bytes. All records must reside in the prime data area; no overflow records are allowed.

EXCLUSIVE CONTROL

CICS/VS optionally provides protection to the user against the concurrent updating of a data base record by two or more transactions. This protection is called exclusive control. The user specifies the exclusive control option in the creation of the FCT.

Several transactions can concurrently update records on the same data set as long as the records being concurrently updated are not in the same sphere of exclusive control. The sphere of exclusive control for BDAM data sets is the physical block. For VSAM data sets, the sphere of exclusive control is the VSAM control interval. For variable-length ISAM (OS) records, the sphere of control is the data set; for fixed-length ISAM (OS) records, the sphere of control is the record key. If a transaction requests a record for update that is within the sphere of control of another record being updated, the second transaction is queued until the first update is complete.

SEQUENTIAL RETRIEVAL

Another optional feature of CICS/VS File Management is the sequential retrieval of records from the data base. This feature is known as browsing. To initiate a browse operation, the user provides either a specific or generic (partial) record reference (key) where sequential retrieval is to begin. Each subsequent GET request by the user initiates retrieval of the next sequential record. The user also can retrieve specified segment sets while in browse mode. The application, while in browse mode, can issue random get for update requests without interrupting the browse operation. The same application can concurrently browse several different data sets and browse the same data set with multiple tasks.

With VSAM data sets, the application can skip forward during the browse operation to bypass unwanted data.

File Management is used by the CICS/VS open/close system service program to support the dynamic opening and closing of the user's data base data sets.

Optional user exits are provided:

- Prior to determining what type of request for file services was issued.
- Prior to providing a requested output service.
- After the File Control Table is searched in response to a request for an input service.
- Upon completion of an input event but before deblocking requested input records.

AUTOMATIC JOURNALING

CICS/VS provides optional automatic logging and journaling facilities for records that are updated, deleted from, or added to a file control data set. Automatic journaling is specified in the File Control Table, by the user, for each data set affected. For a specified data set, a record read for update, a new record added, or an existing record deleted is automatically written to the specified journal.

Following abnormal system termination, data bases defined as recoverable by the user can be restored to their status before being modified by inflight tasks.

In addition to automatic journaling, File Management may perform automatic logging of certain file operations on recoverable files. This logging is written on the CICS/VS system log and on the dynamic log. The information can subsequently be used to restore the recoverable data set as though the current transaction had never run, in the event of either a system or transaction failure.

THE FILE MANAGEMENT MODULES (DFHFCP,DFHFCD)

File Management (DFHFCP and DFHFCD) communicates directly with other CICS/VS functions, the standard access methods, and user-written application programs.

When file services are requested by another CICS/VS module, File Management locates the necessary File Control Table (FCT) entries according to the OPEN/CLOSE/LOCATE Parameter List. It may perform a logical open or close on those entries.

When file services are requested by an application program (through execution of a File Management (DFHFC) macro instruction), fields within the common communication area of the TCA are filled with appropriate entries to communicate with File Management. The File Management interface is determined accordingly; that is, File Management may:

- Request Storage Management to acquire any required storage areas
- Communicate with the standard access methods to request that any required I/O operations be performed (for ICIP files File Management switches to SRB mode before doing this)
- Request Task Management to place the application program (requesting task) in a wait state until I/O operations are completed
- Request Journal Management to perform any automatic journaling required
- Return the address of the appropriate control block to the requesting task in the field TCAFCAA of the task's TCA

The module DFHFCD (in CICS/OS/VS) contains the BDAM and ISAM code.

TRANSIENT DATA MANAGEMENT

Transient Data Management provides a generalized queuing facility enabling data to be queued (stored) for subsequent internal or offline processing. Selected units of information, can be routed to or from predefined symbolic destinations. The destinations are classified as either intrapartition or extrapartition.

INTRAPARTITION DESTINATIONS

Intrapartition destinations are queues of data, held in a direct access (DAM or BDAM) data set, for eventual input to one or more CICS/VS transactions. Intrapartition destinations are accessible only by CICS/VS transactions within the CICS/VS address space. Data directed to or from these internal destinations is called intrapartition data. It can only consist of variable-length records.

The space used by an intrapartition queue is reusable; an option permits the user to indicate, by symbolic name, whether the application program can use the transient data PURGE macro instruction to control the release and reuse of space. If Transient Data Management is not used, the space taken up by a queue continues to grow, irrespective of whether the data has been read, until the user purges the whole queue.

Examples of the data queued for intrapartition processing are:

- Transactions that require processes to be performed serially, not concurrently. An example of this type of process is one in which pending order numbers are to be assigned.
- Data to be used in a data set (file) update that could pass through the queue to allow the data to be applied in sequence.
- Batched input data to be processed asynchronously.

Recovery of Intrapartition Transient Data Queues

Following abnormal system termination, intrapartition destinations defined as recoverable by the user can be restored. Recovery is accomplished by reconstructing the destination control table from log records written automatically by CICS/VS during normal execution. Two types of recovery are possible, physical and logical. If a transient data queue is defined as physically recoverable, its destination control table entry is restored to reflect the status of the queue at the sync point to which the recovery program is backing out. If the queue is defined as logically recoverable, restoration is made as above; however, any changes to the queue made by transactions that were in-flight at the time abnormal termination occurred are backed cut.

EXTRAPARTITION DESTINATIONS

Extrapartition destinations are sequential data sets on tape or direct access devices. Data directed to or from these external destinations is called extrapartition data and can consist of sequential records that are fixed or variable length, blocked or unblocked.

Data can be placed on an extrapartition data set by CICS/VS for subsequent input to CICS/VS or for offline processing. Sequentially organized data created by other than CICS/VS programs can be entered into CICS/VS as an extrapartition data set. Examples of data that might be placed on extrapartition data sets are:

- System statistics
- Transaction error messages
- Customer data, such as cash payments that can be applied offline

INDIRECT DESTINATIONS

Intrapartition and extrapartition destinations can be referenced through indirect destinations. This facility provides flexibility in program maintenance; entries in the CICS/VS destination control table can be changed at environment definition time, giving a destination a new symbolic name, without recompiling existing programs.

AUTOMATIC TRANSACTION INITIATION

When data is sent to an intrapartition destination and the number of entries (PUTs from one or more programs) in the queue reaches a predefined level (trigger level), the user can optionally specify that a transaction be automatically initiated to process the data in that queue.

The automatic transaction initiation facility allows a user transaction to be initiated either immediately, or, if a terminal is required, when that terminal has no task associated with it. The terminal processing status must be such that messages can be sent to it automatically. The destination and the transaction identifications are specified in the Destination Control Table. Through the trigger level and automatic transaction initiation facility, an application program has the ability to switch messages to terminals. Once a task has been initiated, a macro instruction in the application program is executed to retrieve the queued data. All data in the queue is retrieved sequentially for the application program.

Optional user exits are provided:

- After locating the appropriate entry in the Destination Control Table but before writing data in response to an output request.
- After acquiring data in response to an input request.
- Before determining what type of request for transient data services was issued.

TRANSIENT DATA SERVICES

The following services are performed by Transient Data Management in response to CICS/VS macro instructions issued in application programs.

Intrapartition Data Disposition:

Controls and queues data for serially reusable or reenterable facilities (programs, terminals) related to this partition/region.

Intrapartition Data Acquisition:

Retrieves data that has been placed in a queue for subsequent internal processing.

Extrapartition Data Acquisition:

Enters a sequentially organized data set into the system.

Extrapartition Data Disposition:

Writes fixed- or variable-length data in a blocked or unblocked format on sequential devices, usually for subsequent offline processing.

Automatic Transaction Initiation:

Initiates a transaction to process previously queued transient data when a predefined trigger level is reached.

Dynamic Open/Close:

Logically opens or closes specified extrapartition data sets (destinations) during the real-time execution of CICS/VS.

THE TRANSIENT DATA MANAGEMENT MODULE (DFHTDP)

Transient Data Management (DFHTDP) communicates directly with three other functions of the CICS/VS System Management component. These are Task Management, Storage Management, and Program Management. The application program communicates with DFHTDP through use of DFHTD macro instructions.

An application program request for Transient Data services causes a request code to be set in the TCA,signifying GET, PUT, PURGE, FEOV, LOCATE, or CHECK. The destination identification is also placed in the TCAI and control is passed to Transient Data Management

Transient Data Management obtains intrapartition disk map information from the main storage area preceding the Destination Control Table (DCT).

The field TDDCTDID is used to search the DCT for the destination requested. The DCT is checked for OPEN/CLOSE when the reference is to an extrapartition destination.

Task Management is used to wait for access to the DCT and to attach an automatically initiated task not associated with a terminal. If a task is associated with a terminal, the TCTTE is flagged for automatic transaction initiation by Terminal Management.

Transient Data Management communicates with Storage Management to obtain storage for intrapartition I/O areas.

Transient Data Management passes control and the address of the I/O area to the read/write routines of the access methods to perform data set I/O.

If an invalid request for Transient Data services is received, control is passed to Program Management to terminate the task issuing the request.

If restart/recovery is supported within the system, a Deferred Work Element (DWE) is created for each logically recoverable destination when it is accessed. At the end of a logical unit of work (LUW), defined by either an application program DFHSP TYPE=USER request or by Task Management at task termination, CICS/VS Sync Point Management gives control to the DWE processor of Transient Data Management. This DWE processor performs the logical update of the DCT entry and/or the track bit map.

TEMPORARY STORAGE MANAGEMENT

Temporary Storage Management provides the services necessary for an application program to store data temporarily in dynamic storage or on a direct access device. Data is stored and retrieved symbolically, thus facilitating the sharing of data among transactions. Also, if a transaction must be suspended, certain data may have to be saved until transaction processing is resumed.

Temporary Storage Management provides temporary direct access storage that can be used to accumulate data during a transaction that has multiple inputs from the terminal. Temporary Storage Management either puts the data on direct access storage using the virtual storage access method (VSAM), or, if requested, saves the data in dynamic storage.

If the release of data is requested, Temporary Storage Management frees the dynamic storage or direct access storage space that was used for the data.

Temporary Storage Management provides the basic services for the CICS/VS terminal paging function.

TEMPORARY STORAGE MANAGEMENT SERVICES

The following services are performed by Temporary Storage Management in response to CICS/VS macro instructions.

Temporary Storage Put:

Allocates dynamic storage or direct access storage space and saves specified data in this space, using a symbolic identifier provided by the user.

Temporary Storage Get:

Locates data saved in a temporary storage area, puts it into a user-specified dynamic storage location, and, optionally, releases the space the data occupied.

Temporary Storage Release:

Locates data saved in a temporary storage area, removes any control information, and releases the space occupied by the data.

Optional user exits are provided:

- Before writing data in response to an output request.
- Before returning control to an application program after servicing an input request.
- Before determining what type of request for temporary storage services was issued.

THE TEMPORARY STORAGE MANAGEMENT MODULE (DFHTSP)

Temporary Storage Management (DFHTSP) communicates directly with two other functions of the CICS/VS System Management component. These are Storage Management and Task Management. This interface uses the application program's TCA.

An application program requests Temporary Storage services using a DFHTS macro instruction.

Temporary Storage Management communicates with Storage Management to request Temporary Storage unit table extensions, Temporary Storage group identifications (TSGIDs), and main storage for DFHTS TYPE=PUT or TYPE=PUTQ requests to main storage and DFHTS TYPE=GET or TYPE=GETQ requests with no address supplied.

Temporary Storage Management communicates with Task Management to perform a CICS/VS WAIT pending completion of I/O (and suspending and resuming tasks).

Temporary Storage Management interfaces indirectly with VSAM through the Temporary Storage Bit Map. A Request Parameter List (RPL) points to fields in this control block which are passed to VSAM. VSAM communicates with Temporary Storage Management (through Task Management) when the I/O is complete.

Temporary Storage Management analyzes the contents of the RPL to determine the type of completion and any error information.

JOURNAL MANAGEMENT

Journal Management provides facilities for creating and managing special purpose sequential data sets (journals) during the execution of CICS/VS. Journals are intended for recording, in chronological order, any data the user may need to reconstruct events or data sets. For example, journals can be used to record data base updates and additions, system transaction activity, and audit trails. CICS/VS uses the Journal Management facilities in support of the various recovery/restart options provided to its users.

Journal Management consists of the main DFHJCP program, transient subsidiary programs, the Journal Control Table (JCT), CICS/VS macro instructions to request output or retrieval of journal data, a CICS/VS

task for each journal to ensure efficient performance; and one operating system subtask to prevent journal open and close processing, with their possible volume mounting delays, from suspending CICS/VS execution.

JOURNAL MANAGEMENT SERVICES

The following functions are performed automatically by Journal Management without communication from the application program.

Journal Record Identification:

Adds prefix data to each journal record to identify its source. In addition to the user data and a user-supplied unique identifier, the prefix data includes a terminal identifier, a transaction code, and the time of day.

Output Scheduling:

Permits journal data to be written at the maximum rate, subject to host system and storage device limitations. When output requests arrive while the journal is processing another output operation, the waiting requests are consolidated for output in one operation. This operation is initiated as soon as the operation in progress ends.

Volume Management:

Allocates a unique identifier, or label, to each journal volume to assist in the operational control and disposition of multi-volume journal data sets. The label of every off-loading journal tape is reported to the central computer operator.

The following services are performed by Journal Management in response to a specific request from a user application program.

Output Efficiency Option:

Enables the user to minimize the number of journal data set output operations and thus use host system resources more economically. The user specifies the output of a journal record as being immediate or deferred. An immediate-output request causes any outstanding deferred-output records to be included in the one output operation.

Input Capability:

Enables user-written programs to read journal data sets during real-time execution of CICS/VS so that the user can review or reconstruct changes made in the system.

THE JOURNAL MANAGEMENT MODULE (DFHJCP)

Journal Management (DFHJCP) communicates directly with two other CICS/VS management modules: Task Management and Program Management. In

addition, it communicates with the operating system and standard access methods and within its component parts.

Not all portions of Journal Management need be resident in main storage. Some seldom used Journal Management services are performed by separate (transient) modules that are loaded into main storage when required.

CICS/VS provides optional Automatic Journaling facilities for terminal messages and for records that are updated, deleted from, or added to a File Management data set. Automatic Journaling is specified in the File Control Table, by the user, for each data set affected. For a specified data set, a record read for update, a new record added, or an existing record deleted is automatically written to the specified journal.

Journal Management (DFHJC) macro instructions are communicated to Journal Management through the Journal Control Area (JCA). This area must be acquired for a task by issuing a DFHJC macro instruction before any other DFHJC macro instructions are issued by the task. The TCAJCAAD field of the TCA then points to the JCA for the duration of the task.

The normal DFHJC macro instruction expansion sets indicators and addresses in the JCA, before branching to Journal Management.

Program Management instructions are requested by Journal Management when a request for a seldom-used macro service (for example, OPEN, CLOSE, or input) performed by a nonresident (transient) module is received. A DFHPC TYPE=LINK macro instruction is used to request that Program Management load the program. All output and wait requests are serviced by resident code.

Journal Management Open and Close transients use the Journal Management Open/Close Parameter List to communicate requests to an operating system subtask, which then issues the actual Open or Close. The main CICS/VS task remains dispatchable even while a journal open or close is outstanding. Communication is through DOS/VS POST plus CICS/VS Task Management DFHKC TYPE=WAIT. The Journal Management Open/Close Parameter List is located at the start of the Journal Management program and is mapped using COPY DFHJCOCL.

Output requests cause journal records to be built in the appropriate journal's buffer. Journal Management accesses the buffer through pointers in the journal's Journal Control Table Table-Entry (JCTTE). The JCTTE is the repository for all pointers associated with the journal, thus enabling DFHJCP to remain truly reentrant.

The journal buffer is acquired at system initialization and is owned as transaction storage by a separate, never-ending, CICS/VS journal task. Journal tasks, like CICS/VS Terminal Management, run at high priority and enable I/O to be scheduled rapidly and efficiently for common high-usage resources, in this case journal data sets. There is one CICS/VS journal task for each journal data set.

While building a journal record, Journal Management accesses the system area of the caller's TCA for task number and to locate the terminal identification and transaction identification, for inclusion in the system prefix of the journal record. Journal records are mapped using the DSECT DFHJCR.

If I/O initiation is necessary, Journal Management uses the pointers in the JCTTE to access and set up the journal's DTF (DOS) or DCB (OS). In DOS the DTF is mapped using the DSECT DFHJCICA. Journal Management then links to the DASD or Tape IOCS module to initiate I/O. If necessary, Journal Management issues a DFHKC TYPE=RESUME macro

instruction (a Task Management macro instruction documented only for system programming) for the journal task, which in turn issues a DFHKC TYPE=WAIT on the I/O event.

If the user task wishes to wait, Journal Management issues a DFHKC TYPE=WAIT macro instruction on a special logical ECB (LECB). The journal task will POST that LECB when the I/O completes successfully.

The JCT contains a pool of LECBs, from which LECBs are allocated and deallocated by Journal Management as required. The LECB pool is controlled through pointers at the beginning of the JCT, which is mapped using COPY DFHJCTDS.

When there is no outstanding output I/O event for a journal, Journal Management issues a DFHKC TYPE=SUSPEND macro instruction to suspend the journal task.

SYNC POINT MANAGEMENT

Sync Point Management works in conjunction with other CICS/VS components, such as Transient Data Management, Temporary Storage Management, and File Management, to provide the user with the ability to establish points in application program from which it is convenient to restart. Such a point must be one where all changes to the data base have been committed.

Sync Point Management is provided by the module DFHSPP. DFHSPP is invoked by Task Management whenever a task is detached. It can also be called by an application program. Generally, DFHSPP need be invoked by an application program only for a long-running tasks, when it is used to divide the task into shorter units, referred to as logical units of work (LUWs), which better fit recovery requirements.

Deferred Work Elements (DWEs) are created by CICS/VS management modules, are chained off the task's TCA, and represent deferred processing to be done upon completion of a logical unit of work. The module that creates a DWE can insert an entry address of a DWE processor in that DWE. Control is passed to this DWE processor at the end of the task or LUW by the Sync Point program.

Three types of DWEs exist:

- work is to be done
- data is to be logged
- some work is to be done and data is to be logged

A DWE processor pointed to by a work-only DWE can create another DWE, indicating that logging is required.

The Sync Point program examines the DWE chain in the following way:

- Scan the chain and pass control to a DWE processor if work is to be done only.
- Scan the chain and log data as required.
- Scan the chain and pass control to the DWE processor required for a DWE requiring both work and logging (the logging was completed on the previous scan).

A DWE indicating both work and logging to be done implies that the data that is being written to the System Log represents the intention of

the DWE processor. If the system terminates abnormally before the DWE processor has finished its work, the System Log tells CICS/VS modules involved about committed work to be done during emergency restart.

The DWE chain is scanned, and control is passed to DWE processors.

Data is logged, and an end-of-task record is written to the System Log, using Journal Management. All resources enqueued upon through the Task Management DFHKC TYPE=ENQ facility are dequeued. Upon return from a DWE processor, the DWE is freed.

Chapter 7. System Services

A number of ancillary application programs are included in CICS/VS to provide system service functions. Although most of these functions are optional, some of the services that they provide are vital to the successful implementation of a DB/DC system. The system service functions are:

- Sign-on/sign-off
- Master terminal
- Supervisor terminal
- Operator terminal
- System statistics
- Asynchronous transaction processing
- Dynamic open/close
- Time-of-day control
- Terminal test
- Message switching

SIGN-ON/SIGN-OFF

The sign-on/sign-off function is optional. It can be used in various ways if it is selected as an option. For example, only master terminal operators may be required to sign on and sign off. The function can be used to enhance system security by restricting access to some functions.

In sign-on, Program Management is called to load the Sign-on Table (SNT) and then the sign-on security check is made on the name and password parameters supplied by the terminal operator; both must be present, correctly entered, and in agreement with an existing entry in the SNT. If the verification is positive, the Automatic Initiate indicator and operator information are set in the TCTTE, and, if the terminal can receive a reply, the completion message is written.

The sign-off request causes the information set in the TCTTE by sign-on to be cleared. In signing off, the operator can optionally request that the terminal be logically disconnected from CICS/VS. For terminals supported by BTAM, sign-off locates the line for the terminal. It sets the terminal status to unattended, and clears the operator ID. It issues the sign-off status message and, if 'GOODNIGHT' was entered, the No Poll indicator is set. The line condition is changed, the operator class cleared, and then the sign-off program writes the completion message to those terminals capable of receiving it.

MASTER TERMINAL

The overall CICS/VS operation is controlled from the master terminal. The master terminal operator can vary system parameters dynamically, and change the status of each line, terminal, and data set. Individuals within the customer organization can be defined in the Sign-cn Table as having the authority to perform master terminal functions. A designated operator can sign on at any terminal, causing that terminal to have the capabilities of a master terminal.

Master terminal functions include:

- Altering transaction and/or terminal priority
- Disabling and enabling various table entries (PPT, PCT, FCI, DCT)
- Placing terminals in service or out of service
- Dynamically listing active tasks
- Purging active tasks
- Dynamically opening/closing selected data sets
- Requesting a CICS/VS formatted dump
- Switching to an alternative dump data set

The Master Terminal program (DFHMTP) consists of seven modules: DFHMTPA, DFHMTPB, DFHMTPC, DFHMTPD, DFHMTPE, DFHMTPF, and DFHMTPG.

The Master Terminal program is invoked by an operator entering CSMT, CSST, or CSOT at a terminal, which may be designated as a master or supervisory terminal, or may be a normal unspecialized terminal. The transaction identification may be followed by a series of keywords describing the services to be performed. If the keyword CANCEL is entered anywhere in the original message or subsequent entries, the Master Terminal program is terminated immediately. If, while trying to perform a requested service, the Master Terminal program discovers that insufficient information has been entered, additional information is solicited from the requesting terminal.

All transactions with transaction identifications of CSMT, CSST, or CSOT are placed in a TIOA accessible to DFHMTPA. This input is scanned to determine whether sufficient keywords are present to specify the requested service fully. Otherwise, additional information is requested from the input terminal. A keyword indicator is set in the requesting task's TWA for each keyword entered. Control is transferred to the DFHMTP module that will provide the requested service. If time or runaway task services are requested, DFHMTPA responds to the request.

DFHMTPB responds to all requests for file, cushion, maximum task, negative poll delay, and trace services. File Management is used to locate file entries in the File Control Table (FCT).

DFHMTPC responds to all requests for terminal related services.

DFHMTPD responds to all requests for opening, closing, and switching of Dump data sets. The Dynamic Open/Close program (DFHCCP) is used to perform the actual opens andr closes.

DFHMTPE responds to all requests for Transient Data, stall, trigger level, and NEWCOPY services. Transient Data Management is used to locate destinations in the Destination Control Table (DCT).

DFHMTPF responds to all requests for line, control unit, terminate task, and task list services. Task Management is used to schedule a task for termination.

DFHMTPG responds to all requests for transaction and program services.

DFHSTP responds to requests for system shutdown.

SUPERVISOR TERMINAL

Individuals within the organization can be defined in the Sign-on Table as having authorization to perform supervisory functions. A designated individual can sign on at any terminal, causing that terminal to be designated as a supervisory terminal. Using the command CSST, supervisors can change the service status or the processing status of of any terminals under their supervision. A terminal can be placed either in service or out of service. Its processing status can be such that it can initiate transactions and receive messages on request, receive messages automatically, receive messages only, or send messages to CICS/VS only.

Supervisor requests are checked for validity and processed by the Master Terminal program.

OPERATOR TERMINAL

Terminal operators who are neither supervisors nor the master terminal operator are only able to control the service status and the processing status of their own terminals, using the command CSOT.

Operator Terminal requests are checked for validity and processed by the Master Terminal program.

SYSTEM STATISTICS

System statistics are maintained by CICS/VS management programs during the execution of CICS/VS. These statistics can be displayed during the day, in part or in their entirety, on the request of any terminal operators whose security codes allows them access to such information. The statistics are printed automatically when the system is terminated normally. Statistics totals can be reset when they are read out, if requested by the master terminal operator.

System statistics are transmitted to the user-specified destination as variable-length, unblocked records (maximum block size equal to 136) when the system is terminated normally, or when requested by a terminal operator. The default destination is the system log destination (CSSL).

In addition to requested system statistics there are automatic system statistics. Requested statistics are obtained at the time of the request. In contrast, once automatic statistics mode is initiated,

statistics are obtained automatically at periodic intervals. Requested statistics cannot be obtained while CICS/VS is in automatic statistics mode.

System statistics are maintained by the various CICS/VS management programs during execution of CICS/DOS/VS. These statistics can be written out on the request of any terminal operators whose security code allows them to have such information.

System statistics are transmitted to a default destination or to a user-specified destination (sequential output device) as variable-length, unblocked records (maximum block size is 120). The default destination is the control system system log destination (CSSL).

The statistical information maintained by CICS/VS includes:

- Number of tasks in system for any time period
- Number of tasks initiated
- Number of storage acquisitions
- Number of times storage cushion is released
- Number of times storage request is queued
- Number of times storage queue is established
- Maximum number of requests in storage queue
- Number of times a program is used
- Number of READ requests per data set
- Number of WRITE update requests per data set
- Number of WRITE add requests per data set
- Number of deletes from VSAM data set
- Number of WRITEs (per data set) to extrapartition data set
- Number of WRITEs (per data set) to intrapartition data set
- Number of input message per terminal
- Number of output messages per terminal
- Number of transmission errors per terminal
- Number of transactions
- Number of transaction errors
- Number of polls issued per line
- Maximum number of Temporary Storage control intervals used (for auxiliary storage)
- Maximum number of PUTs to Temporary Storage main storage and auxiliary storage
- Maximum number of Temporary Storage unit tables used
- Maximum number of Temporary Storage group identifications used
- Temporary Storage main storage requirements
- Number of records written per journal identification

Through the use of CICS/VS Journal Management, the user can create and log transaction data to sequential data sets, called journal data sets. The user describes each journal data set when defining his system. During CICS/VS execution, an application program issues CICS/VS macro instructions that cause Journal Management to store transaction data in that data set when the transaction is completed.

If DL/I data bases are used, the DL/I logging function produces a data set which is an audit trail of changes made to DL/I data bases. A scheduling or termination record is logged to this data set when a transaction that alters a DL/I data base is scheduled or terminated. DL/I provides utilities that can be used to produce reports from the statistics that it collects.

ASYNCHRONOUS TRANSACTION PROCESSING

The optional Asynchronous Transaction Processing facility (ATP) allows transactions and the data associated with those transactions to be batched for asynchronous processing. This means that the transactions within a batch are not processed until the entire batch is completely entered into CICS/VS. Then, while the batched transactions are being processed, the terminal operator may proceed to enter other transactions. When the batched transactions have completed processing, the terminal operator may request that the output, if any, be sent to the terminal that originated the batch, or to some other terminal. ATP is supplied by CICS/VS to provide a batch data collection capability from pre-SNA, remote batch terminals. Asynchronous transaction processing is not available for terminals connected through VTAM, nor, in general, is it necessary for such terminals.

When all input batches have been transmitted, the transactions within the batch are then processed by application programs based upon their respective transaction codes. Any error messages are directed by the editing program to transient data for later transmission back to the terminal.

When the batches have completed processing, the terminal operator may then request that the output, if any, be sent to the terminal that originated the batch, or to a different terminal. Depending upon the amount of processing to be carried out on transmitted batches, the batch terminal may be disconnected from the transmission line by the user until output is available to be transmitted back to it.

The ATP facility is designed specifically for handling input from pre-SNA batch terminals such as the 2770, 2780, and 3780. ATP can also be used with some interactive terminals, such as the 2741. ATP is not available with the following devices:

The Asynchronous Transaction Control program (DFHATP) controls the initiation and data handling of all asynchronous tasks submitted as part of a batch. The batch consists of one or more CICS/VS transactions, along with any associated data, entered into CICS/VS through the Asynchronous Transaction Input Processing programs (DFHRD1 and DFHRD2), which are invoked by the transaction code CRDR.

When an entire batch has been submitted, a Batch Control Area is created and put on a chain addressed through the CSA, then the transactions are executed, asynchronously with other possible terminal activity, by the originating terminal. When all transactions have been processed, the output of the batch can be transmitted back to a terminal, depending on how the batch was entered, by the Asynchronous Transaction Output Processing programs (DFHWT1 and DFHWT2). If not transmitted automatically, the output remains queued until it is requested by the originating terminal or an alternate terminal. A batch may also be deleted by the Asynchronous Queue Purge program (DFHAQP).

ATP is executed as a unique CICS/VS task, with its own TCA, and may be resident and active only when one or more batches exist within CICS/VS. ATP is composed of the ATP input processor (DFHRDR), the ATP output processor (DFHWT1, DFHWT2), and the ATP control program (DFHATP).

ATP communicates directly with CICS/VS System Management functions, and communicates indirectly with the user-exit program through the Terminal Input/Output Area (TIOA) and the Transient Data intrapartition queue. Internal communication within ATP is also indirect, using the same control blocks.

ATP queues the input data onto a Transient Data intrapartition queue. Transient Data Management services are requested by DFHTD TYPE=PUT requests. Storage Management is called to obtain and release the TDOA. When an entire batch has been read, the Asynchronous Transaction Control program (DFHATP) is either attached (by a Task Management DFHKC TYPE=ATTACH macro instruction) or marked "ready to run." If a batch is to be deleted, the Asynchronous Queue Purge program (DFHAQP) is attached to perform the purging of data from the Transient Data intrapartition queue.

ATP interprets CWTR requests for output and returns any messages to the requesting terminal.

The BCA chain is searched to locate any batches requested by the CWTR command. A Write Request Element (WRE) is built for each terminal that is to receive output, and Storage Management services requested by a DFHSC TYPE=GETMAIN, to obtain a storage area for the WRE. Before the output operation, a Terminal Input/Output Area (TIOA) is acquired by a Storage Management.

ATP interacts with any user-exit routines (for more about these routines, see the CICS/VS System Programmer's Reference Manual). It then retrieves the data from a Transient Data intrapartition queue, using Transient Data Management.

At the end of a CWTR operation initiated by a WRE, the WRE is removed from the WRE chain and its storage released by using a Storage Management DFHSC TYPE=FREEMAIN. After all WREs have been removed from a BCA, and if SAVE was not specified, the BCA is removed from the BCA chain and its storage released in a similar way. ATP interfaces with the Asynchronous Queue Purge program (DFHAQP), which is attached by a Task Management DFHKC TYPE=ATTACH macro instruction.

If RELEASE was specified in the CWTR command, the Asynchronous Transaction Control program (DFHATP) is attached, if it is not already present. The storage is released, and the task is terminated.

ATP is attached by either a CRDR transaction (RDR) or Terminal Management in response to a terminal command (CATP or an application program read or write request).

Any DFHTC TYPE=READ or TYPE=WRITE request issued by an application program causes ATP to extract data from the Transient Data intrapartition input queue, or to place data onto the Transient Data intrapartition output queue, respectively. All rules that apply to the handling of Terminal Input/Output Areas (TIOAs) when a task is directly connected to a terminal also apply to tasks being run asynchronously. For this purpose, each task initiated by ATP is given the address of the dummy TCTTE, which is available in the BCA.

Whenever a BCA is to be deleted, the Asynchronous Queue Purge program (DFHAQP) performs the purging of data from the Transient Data queues used to process batches.

The initial application program for each valid transaction code is attached by DFHATP using Task Management. Messages describing any errors detected during application program processing of the batch are queued by CICS/VS. These error messages are transmitted back to the remote terminal on request, to permit batch error correction and resubmission of corrections if required.

DYNAMIC OPEN/CLOSE

Dynamic Open/Close allows the user to open and close data sets as often as necessary while CICS/VS is running. This makes it possible for the user to defer the opening of data sets at system initialization, and open and close them later as they are needed.

The Dynamic Open/Close program (DFHOCP) enables the user to open and close Dump data sets, Transient Data extrapartition data sets, and File Management data base data sets. Dynamic Open/Close may be invoked by using the CICS/VS Master Terminal program, or by using DFHOC macro instructions in an Assembler-language application program.

The DFHOC macro expansion issues a Program Management DFHPC TYPE=LINK macro instruction to transfer control to the Dynamic Open/Close program.

Dynamic Open/Close communicates with:

- Storage Management, in response to all requests for services
- Dump Management in response to requests for open, close, or switch of Dump data sets
- File Management to open or close data base data sets
- Transient Data Management and Program Management to open or close extrapartition data sets

Dynamic Open/Close returns to the requesting program by a Program Management DFHPC TYPE=RETURN macro instruction.

TIME OF DAY CONTROL

The optional Time of Day Control program (DFHTAJP) makes it possible for the user to operate CICS/VS continuously, for more than twenty four hours. When the time of day maintained by the operating system is changed by the operating system (for example, when the clock is reset to zero at midnight), CICS/VS recognizes the situation where a negative change in the time of day has occurred and adjusts the expiration times maintained by CICS/VS, and then resets its time of day to that maintained by the operating system.

The clock reset service of Time Management is called to find the current clock value and adjustment value. The adjustment time is calculated (24 hours being used if a difference of 23 hours 30 minutes or more is detected between the current and prior times). Time Management is called again to set the new clock values and to update the date and time slots in the CSA.

TERMINAL TEST

The terminal test program (DFHFEP) is a CICS/VS system service function primarily designed for an IBM field engineer to use when installing new terminals. When CICS/VS is running, this program, invoked by the transaction code CSFE, transmits all printable characters to the requesting terminal. In addition, the program can be used to echo a message; that is, it repeats exactly what is keyed at the terminal.

The routine prepares for device dependent conditions. It then issues a Storage Management FREEMAIN, followed by a GETMAIN for storage for the ENTER message, which it writes using Terminal Management WRITE, READ, and WAIT macros. Finally, if 'print' was requested the character set is printed; if 'end' was requested then the completion message is issued; otherwise the input is echoed.

MESSAGE SWITCHING

CICS/VS provides the user with a general purpose message switching capability. Messages can be routed to one or more terminals by specifying the transaction code CMSG, the message text, and optional parameters that identify the receiving terminals.

The Message Switching program (DFHMSP) runs as a task under CICS/VS. A terminal operator requests activation of this task by entry of the transaction identification CMSG (or another installation-defined four-character transaction identification), followed by appropriate parameters. Once initiated, Message Switching interfaces with CICS/VS Basic Mapping Support (BMS) and CICS/VS management functions. Although Message Switching appears conversational to the terminal operator, the message switching task is terminated after each terminal response. Through the entry in the PCT which holds the the address of the program DFHMSP, the CICS/VS user can define the transaction identification (which must be four characters long) for message switching. Conversation is forced, if continuation is possible, by effectively terminating the transaction with a DFHPC TYPE=RETURN,TRANSID=XXXX, where XXXX is the transaction identification taken from the task's PCT entry.

If the first four characters of the TIOA do not match the transaction identification in the task's PCT entry, then this task must have started as part of a conversation, because a previous task set up the next transaction identification. A 'C' immediately following the transaction identification also forces continuation. In such a case, information has been stored in, and has to be retrieved, from Temporary Storage (using a record key of one-byte X'FC', four-byte terminal identification, and three-byte C'MSG') to allow the task to resume where it left off.

The operands in the input TIOA are processed and their values and status are stored in the TWA. If a ROUTE operand specifies one or more Terminal List Tables (TLTs) for a standard routing list, a DFHPC TYPE=LOAD macro instruction is issued to request the Program Management to load the requested TLT(s).

Message Switching requests storage areas for:

- Building route lists (one or more segments, each of which has room for the number of destinations specified by MSRTELNG, an EQU within the program)
- Constructing a record to be placed in Temporary Storage
- Providing the message text to BMS if message parts from previous inputs exceed the current TIOA size, a message is completed in the current TIOA but has parts from previous inputs, or a heading has been requested but the message in the current TIOA is too close to TIOADBA to allow the header to be inserted.

Message Switching requests BMS routing functions by using the DFHBMS TYPE=ROUTE macro instruction. The message text is sent using DFHBMS TYPE=TEXTBLD, and completion of the message is indicated by DFHBMS

TYPE=PAGEOUT. BMS returns the status of destinations and any error indications in response to the DFHBMS TYPE=CHECK macro instruction.

Message Switching interfaces with BMS using DFHBMS TYPE=(EDIT,OUT) and with Terminal Management using DFHTC TYPE=WRITE (for 3270 only), in providing responses to terminals. These may indicate normal completion, signal that input is to continue, or provide notification of input error.

Like any other task, Message Switching has a Task Control Area (TCA) in which values may be placed before issuing CICS/VS macro instructions and from which any returned values can be retrieved after an operation. All values for the DFHBMS TYPE=ROUTE macro instruction are placed in the TCA because they are created at execution time. The TWA is used for storing status information (partly saved in Temporary Storage across conversations) and space for work area. The DFHMSP module is reentrant.

Chapter 8. System Monitoring

The System Monitoring component consists of two functions that provide diagnostic traces and dumps.

TRACE MANAGEMENT

The optional trace function provides a trace table containing entries that reflect the execution of various CICS/VS macro instructions by application programs and CICS/VS management programs. The trace facility is especially useful in program testing and debugging. The trace function, if generated, can be turned on or off by the Master Terminal operator.

The Trace Management module (DFHTRP) contains routines which enable the user to turn tracing on and off, and tc record a specified item in the trace table. The address of the trace table is held in the CSA. The trace table starts with fields giving the address of a header area, which in turn holds the address of the most recently recorded entry. The format of the Trace Table is given in the CICS/VS Level Application Programmers Reference Manual (Macro Level).

The command interface program also provides a trace of EXEC commands. This is described in the CICS/VS Application Programmer's Reference Manual (Command Level).

AUXILIARY TRACE MANAGEMENT

This optional function writes auxiliary trace records to a sequential data set. The format of the records is: time-of-day, followed by the standard trace entry. Information has been added to supplement that provided by standard CICS/VS Trace Management, so that a complete trace of a CICS/VS execution can be obtained. These entries are triggered by the following events:

- Task dispatched
- Task created (indicates that a TCA has been built and a task number assigned)
- Task terminated (indicates release of a TCA)

The information in the auxiliary trace table may be used for problem determination and performance tuning. The auxiliary trace function, if generated, may be turned on or off through the CICS/VS master terminal facility.

If the trace has been switched on by a DFHTR macro instruction, then standard entries will be recorded in the trace table whenever CICS/VS macro instructions of the following types are issued by an application program or a CICS/VS system program:

DFHKC (Task Management)	DFHTD (Transient Data Management)
DFHSC (Storage Management)	DFHTS (Temporary Storage Management)

DFHPC (Program Management)
DFHIC (Interval Management)
DFHDC (Dump Management)
DFHFC (File Management)
DFHSP (Sync Point Management)
DFHJC (Journal Management)
DFHBMS (Basic Mapping Support)
DFHBIF (Built-In Functions)
DFHDI (Data Interchange)

trace entries are also produced when the following programs are entered:

CICS/VS-DL/I Interface
Dynamic Backout Program
EXEC Interface Routine

These standard entries in the trace table are generated by special DFHTRACE macro instructions in the relevant CICS/VS management modules. Entries may also be generated by the execution of the DFHTR TYPE=ENTRY macro instruction. The trace is turned on and off by setting a flag, which is done by the application program itself (in the inserted macro expansion) in the case of an Assembler-language program, or by Program Management in the case of an ANS COBOL or PL/I program. Auxiliary Trace Management writes the trace entries, including time-stamp values, to the Auxiliary Trace data set.

DUMP MANAGEMENT

The Dump Management module (DFHDCP) provides a diagnostic facility to help in analysis of programs that are undergoing development or modification. This facility locates and makes available images of specified information to a sequential data set for subsequent printing by the CICS/VS dump utility program. Requests for storage dumps are communicated to the Dump Management through CICS/VS DFHDC macro instructions.

The Task Control Area (TCA), and Common System Area (CSA) are always dumped. In addition the following areas may be dumped: Program check registers and PSW, trace table, segment storage, transaction storage, facility control area, terminal storage, program and register save areas, and system control tables.

Dump Management records dumps of transactions on sequential data sets that are located either on magnetic tape or on direct access storage. CICS/VS provides the capability to open or close the active dump data set during CICS/VS execution.

Optionally, the user can define two dump data sets (DFHDMPA and DFHDMPB), alternating between them during the execution of CICS/VS. If DFHDMPA is opened by System Initialization and if a request is issued, through the CSMT command, to use the alternate dump data set, DFHDMPA is closed and DFHDMPB is opened. Another request to use the alternate dump data set causes DFHDMPB to be closed and DFHDMPA to be opened.

When a storage dump is in progress, Dump Management delays the processing of subsequent dump requests until the current dump is completed. Dump Management writes specified areas of storage to tape or disk (to be printed later by the Dump Utility program). DFHDCP may be called by either an application program or a CICS/VS management program. The DFHDC macro instruction sets the type of request switch, number of bytes to be dumped, storage address, and dump identifier in the requesting program's TCA.

Dump Management communicates with Time Management, Trace Management, and Program Management to set runaway task control, request and release

a lock of the dump facility, request a trace, and reset and test runaway task control.

Output is written to a sequential data set on tape or disk. Records have undefined length and are written as a continuous string of data. A single record may be a CICS/VS system control table, one TCA, or an entire program. Each record has an identification field preceding it to identify the record for the Dump Utility program which prints the dump.

Chapter 9. System Reliability

The system reliability component consists of functions that help the user to deal with error conditions. These functions are performed online.

SYSTEM RECOVERY MANAGEMENT

The System Recovery program (DFHSRP) combines program interrupt handling with operating system abnormal termination handling to prevent uncontrolled termination of CICS/VS. System Initialization establishes the necessary operating system linkage by issuing STXIT macro instructions (under DOS/VS) or the SPIE and STAE/ESTAE macro instructions (under OS/VS).

The System Recovery program is given control by the operating system when a program check or abnormal termination recognized. The System Recovery program determines what action is to be taken, on the basis of information in the System Recovery Table (SRT), and the type of error. If a CICS/VS task is to be abnormally terminated, control is given to Program Management.

If a program check occurs, the transaction that has control of CICS/VS is abnormally terminated with a CICS/VS dump (if Dump Management is included); other transactions are allowed to continue processing. If an operating system abnormal termination occurs in the CICS/VS partition or region, CICS/VS scans the SRT. The SRT lists the abnormal termination codes for which user-written or CICS/VS-supplied routines are to be executed. If the code is found, control is given to the designated routine.

Completion of the recovery routine is followed by termination of the task that was in control when the abnormal condition arose. A CICS/VS dump is taken if Formatted Dump is present. The user can choose to terminate CICS/VS. If the code is not found in the SRT, keypoint data can be written and CICS/VS will be terminated.

EMERGENCY RECOVERY/RESTART

CICS/VS provides an emergency recovery/restart facility when system execution is interrupted before controlled shutdown can be performed. This recovery/restart facility provides the following major functions:

- Recovery of intrapartition transient data queues
- Recovery of temporary storage
- Recovery of interval control data
- Recovery of data base data sets and DL/I data bases
- Recovery and Resynchronization

DYNAMIC TRANSACTION BACKOUT

The Dynamic Transaction Backout program (DFHDBP) is invoked by Program Management. The program starts by processing Deferred Work Elements (DWEs), scanning the chain of DWEs from the TCA, as follows:

- Temporary Storage Management and Transient Data Management DWEs are modified to indicate backout so that the relevant management modules perform the backout function.
- BMS messages corresponding to DWEs are purged by calling a BMS routine.
- File Management DWEs are processed to release any FIOAs and VSWAs for VSAM.
- DL/I DWEs are processed to unlock code in Sync Point Management.

The program, having worked through the DWEs, then chains back through the dynamic log. According to the type of record in the log it takes the following actions:

- Chain records - the program gets the previous record in the log.
- Overflow record - the program gets a buffer (using Temporary Storage Management) and reads the record into it.
- First Input Message Record - the program passes the record to a user exit.
- File backout record - GET UPDATE and PUT NEW File Management operations are logged. For GET UPDATE a copy of the the record before the operation is logged; for PUT NEW the identification of the added record is logged. The Dynamic Transaction Backout program restores the copy for GET UPDATE. For PUT NEW the record is deleted if possible; if the access method is such that deletion is impossible then a user exit is called, and if the record does not exist then no backout is needed.
- DL/I Backout records - The program sets up the DL/I environment and calls the DL/I backout routine. Any DL/I error message is sent to CSMT and also causes a call to a user exit.

The Dynamic Transaction Backout program finishes by transferring control to the Abnormal Condition program.

RECOVERY UTILITY PROGRAM

The Recovery Utility program (DFHRUP) is invoked by System Initialization in the event of an emergency restart. The object is to restore the system to a point at which the data bases were self-consistent. Before calling the Recovery Utility program, System Initialization restarts the DCT, PPT, PCT, TCT, FCT, and CSA. (This is a warm start or a cold start, according to user options.) System Initialization also provides a cold start for Transient Data and Temporary Storage and builds their bit maps to indicate that all tracks or control intervals are empty.

During an emergency restart, the System Log is automatically repositioned after the last record written during the previous execution. The Recovery Utility program calls Journal Management to read this data set backwards in order to process system recovery data and to collect user recovery backout data. The backward scan is completed and more user records cannot be collected when the following conditions are met:

- At least one complete activity keypoint, has been retrieved.
- The start of each Logical Unit of Work (LUW) which was in-flight at system termination has been reached.
- All in-doubt committed output messages have been recovered

During the backward scan, the Recovery Utility program uses the Keypoint program to output to the Restart data set the following data:

- Records output to the System Log by tasks (LUWs) that did not complete processing (in-flight tasks) before the system abnormally terminated. These records follow the standard Journal Control Record Layout, and they are as follows:
  - Records automatically logged by File Management for data sets with the specification LOG=YES in the FCT.
  - Records automatically journaled to the System Log by File Management, according to the user-specified option in the FCT.
  - User-journaled records to the System Log that were output by in-flight tasks. User-journaled records with the high-order bit set ON in the JTYPEID which are encountered during the backward scan, are copied over to the Restart data set regardless of the status of the task (in-flight or complete). User-written activity keypoint records should have an identification as stated above in order to be accessible from the Restart data set.
- Initial input and final output messages per LUW are logged by Terminal Management for terminals with the Protect option group, specified in the Program Control Table (PCT).
- All input/output messages are recorded by Journal Management, as specified in the MSGJRNL=operand in the PCT.

ABNORMAL CONDITION

The Abnormal Condition program (DFHACP) is used by Program Management to analyze abnormal conditions when they occur and to take appropriate action. For example, the terminal operator is notified that a task has been terminated and appropriate messages are logged to the transient data master terminal destination (CSMT). Statistics are gathered for the number of transaction errors.

Task abnormal conditions are detected by CICS/VS management programs and are often due to an application program destroying system control information. When this happens, the task is terminated, the terminal operator is informed of the error, and the error is logged at destination CSMT (computer system master terminal).

Operator errors may occur, such as invalid transaction identifications, security key violations, or failure of an operator to sign on the system before attempting to communicate with CICS/VS. When

this happens, the terminal operator is notified, and the error is logged at destination CSMT.

The Abnormal Condition program is invoked by Task Management whenever an invalid transaction code is detected, or there is a security violation or a failure to sign on.

The Abnormal Condition program is invoked by Program Management whenever a task is abnormally terminated. The Sign-On Table (SNT) is loaded to obtain the operator identification (if present) for error messages. The Abnormal Condition program returns to Program Management after the error message has been issued. When a task is abnormally terminated because of a stall purge condition, the stall purge count is increased by one and the transaction identification (from the Program Control Table) is included in the error message.

The Abnormal Condition program calls Storage Management to obtain and release Terminal Input/Output Areas (TIOAs) and Transient Data Output Areas (TDOAs) for writing error messages.

If a user-written Program Error program (DFHPEP) is provided, the Abnormal Condition program links to it (using a conditional DFHPC TYPE=LINK instruction). All user-written PEPs communicate their requests through the TCA. The CSA contains the addresses of programs, and the Terminal Control Table terminal entry and line entry (TCTTE and TCTLE) hold the status of the terminal and the line.

Error messages are written to Transient Data destination CSMT.

The Abnormal Condition program communicates with Dump Management to initiate a complete dump, identified by the code AACA.

PROGRAM ERROR PROGRAM

The user who wishes to provide corrective action in response to a programming error can do so by coding a Program Error Program (DFHPEP). If provided, this program is linked to by the Abnormal Condition Program (DFHTACP) whenever a task terminates abnormally. The only exception to this procedure is when CICS/VS deliberately terminates a task to alleviate a stall situation.

The user can perform any kind of corrective action within PEP. CICS/VS provides the option of disabling the transaction code associated with the program in error, thus preventing the recurrence of the error until it can be corrected.

TERMINAL ABNORMAL CONDITION PROGRAM (BTAM, GAM)

The Terminal Abnormal Condition Program (DFHTACP) is used by Terminal Management to analyze any abnormal conditions. Appropriate action is taken with regard to terminal statistics, line statistics, terminal status, and line status; the task (transaction) can be terminated. Messages are logged to the transient data master terminal destination or terminal log destination. A link is provided to the user-written terminal error program to allow the user to attempt recovery from transmission errors and to allow the task to continue processing.

TERMINAL ERROR PROGRAM (BTAM, GAM)

Users who wish to provide their own corrective action whenever a terminal I/O error occurs can do so by coding a Terminal Error Program (DFHTEP). If provided, this program is linked to by the Terminal Abnormal Condition Program whenever an unrecoverable I/O error occurs on a terminal. CICS/VS also provides a sample DFHTEP.

The user can perform any kind of corrective action within DFHTEP; CICS/VS provides options that either place the terminal out of service or retry the I/O operation.

NODE ABNORMAL CONDITION PROGRAM (VTAM)

The Node Abnormal Condition Program (DFHZNAC) is used by Terminal Management to analyze the abnormal condition. The program then takes appropriate action with regard to terminal status and statistics. The task (transaction) can be terminated. Messages are logged to the transient data master terminal destination or terminal log destination. The program provides a link to the appropriate Node Error Program.

NODE ERROR PROGRAM (VTAM)

The user will need to provide a Node Error Program (DFHNEP) for VTAM-supported terminals. CICS/VS provides a sample node error program which assists the user in three ways:

- by providing a general environment within which it is easy for users to add their own error processors.
- by providing the fundamental error recovery actions for a VTAM 3270 network. These actions are consistent with those provided in the sample terminal error program for BTAM 3270.
- by serving as the default NEP where the user selects a NEP at System Initialization.

KEYPOINT PROGRAM

The Keypoint program (DFHKPP) is activated for one of two purposes:

- Warm Keypointing:

  Collecting and recording data from system tables and control blocks, and writing that information to the Restart data set for use by System Initialization in a subsequent warm start of CICS/VS

- Activity Keypointing:

  Collecting and recording data from system tables and control blocks, and writing that information to the System Log for use by the Recovery Utility program in a subsequent emergency restart of CICS/VS

WARM KEYPOINTING

The Keypoint program is linked to by System Termination when CICS/VS is closed down in response to a user request for termination.

The System Recovery program may generate a link to the Keypoint program when an unrecoverable error condition precludes further execution of CICS/VS.

Information collected by the Keypoint program is written onto the Restart data set (DFHRSD), which is a direct access data set with user-specified block size preformatted by System Initialization.

The collected information consists of:

- PPT, PCT, and FCT - the entire tables
- TCT - the non-switched TCTTEs
- DCT - the intrapartition entries and the bit map
- TSUT - the auxiliary destination identifications, queue counters, RBAs, and the bit map
- Interval Control Elements (ICEs) and Automatic Initiate Descriptors (AIDs) - the entire control blocks
- Batch Control Areas (BCAs) - the entire control blocks and their associated Write Control Elements (WREs)
- CSA - certain fields, such as time intervals and maximum task values

When all the data has been recorded, a time-stamped control record is written to DFHRSD. This record contains DASD addresses of the data and is used by System Initialization at warm-start time. Control is then returned to the caller, which is either System Termination or System Recovery.

ACTIVITY KEYPOINTING

The Activity Keypoint program (DFHAKP) is invoked by transaction CSKP when activity keypointing is required. It invokes the Keypoint program to perform the keypointing. When control is returned to Activity Keypointing, an exit can be taken to a user-written routine (DFHUAKP) for further processing.

The need for activity keypointing is signalled when an activity keypoint frequency count is reached during Journal Management logging of activity on the System Log (Journal ID 1).

Journal Management attaches the task associated with transaction identification CSKP. CSKP invokes the Activity Keypoint program which passes control to the Keypoint program, which gathers the keypoint information in buffers and calls Journal Management to record this information on the System Log. The Keypoint program returns to Activity Keypointing, which writes a time stamp to the master terminal (CSMT).

Chapter 10. System Support

The system support component consists of several functions that are required in order to run CICS/VS. Most of the support functions are performed offline.

SYSTEM GENERATION

System generation allows the CICS/VS user to define and structure CICS/VS to meet his teleprocessing needs. This process allows him to select only those functions required by his applications.

System generation is a two-stage operation. The input to Stage I consists of user-prepared CICS/VS system generation (DFHSG) macro instructions, which are assembled to produce a job stream that is used as input to Stage II. The Stage II input job stream is comprised of jobs that assemble (or collect pre-assembled versions of) CICS/VS management and service programs, and link-edit all modules to the CICS/VS Program Library.

ENVIRONMENT DEFINITION

The operating environment required for real-time processing is defined and controlled by user-generated control tables and service tables. Because CICS/VS allows the user to maintain a number of versions of tables (and programs) through the use of suffix characters, the real-time environment is highly flexible and easily managed. Environment definition is described in more detail in Chapter 3 of this manual.

CICS/VS is dependent on the user-created system tables, which describe the user's data base/data communication environment and the treatment to be given to elements of that environment. Information regarding the user's terminals, data sets (permanent and temporary), programs, and transactions is contained in these tables.

Each table is created separately and may be re-created at any time prior to system initialization. The tables are created independently of system generation, but some of the tables are required for the system to be operational. More than one system table of each type, other than the Sign-on Table, can be maintained and be available for use. This allows the user to maintain special tables for testing in addition to tables for normal, routine operation. Each system table is prepared by assembling the appropriate macro instruction with its associated operands (see the CICS/VS System Programmer's Reference Manual for details of these macro instructions). The output of each macro instruction assembly contains the linkage-editor control cards required to link-edit the table into CICS/VS.

SYSTEM INITIALIZATION

The System Initialization program (DFHSIP) is used to start CICS/VS. The process of starting CICS/VS includes the following major steps:

1. Establish initialization parameters; provide the user with the ability to configure a CICS/VS nucleus through use of the system initialization table and to specify various system control parameters that affect system performance.

2. Load selected CICS/VS management programs and tables to form the CICS/VS nucleus.

3. Open CICS/VS system data sets and user data sets.

4. Establish the dynamic storage boundaries for the CICS/VS storage control program.

5. Optionally, reinitialize the CICS/VS system from information obtained during the prior execution (warm start).

6. Optionally, transfer control to one or more programs defined in a Program List Table.

7. Load the resident application programs.

8. Transfer control to an entry point in the CICS/VS nucleus, which in turn branches to Terminal Management to begin polling.

9. Optionally, restore recoverable resources following an abnormal termination (emergency restart).

The main storage occupied by the system initialization program becomes part of the dynamic storage area.

RESTART AT SYSTEM INITIALIZATION

System initialization provides three classes of restart:

COLD
: Complete reinitialization of CICS/VS and system data sets, ignoring any previous system activity.

WARM
: This optional restart process reinitializes CICS/VS to the status that existed at the previous system termination. This type of restart assumes that the previous termination was normal, that the system was quiesced before termination, and that a warm keypoint was taken during that termination of CICS/VS.

EMER
: The emergency optional restart process restores the system, using information recorded during the previous execution of the system, to some point before the interruption.

Selection of startup options can be made in the System Initialization Table specification (COLD or WARM) or in the override parameters (COLD, WARM, or EMER).

RESTART DATA SET

The Restart Data Set is a DAM file, used by the Keypoint program (DFHKPP) to save certain system information at system termination time

so that a warm start can be initiated later. System Initialization can warm start the following CICS/VS control information:

Program Control Table (PCT)
Processing Program Table (PPT)
Terminal Entries (nonswitched)
File Control Table (FCT)
Selected areas from the Common System Area (CSA)
Destination Control Table (DCT) Intrapartition Entries
Transient Data Intrapartition space allocation bit map
Identifications and Relative Byate Addresses for Temporary Storage auxiliary destinations/queues
Temporary Storage space allocation bit map
Interval Control Elements (ICE) and Automatic Initiate Descriptors (AID)
Batch Control Areas (BCA) and Write Request Elements (WRE) for ATP

Under DOS/VS System Initialization receives control from DOS/VS Job Control. Parameters may be passed to System Initialization through SYSIPT. (These parameters are documented in the CICS/VS System Programmer's Reference Manual.)

The Restart data set is read by System Initialization to give a warm start to CICS/VS, if warm start of the system is requested.

When emergency restart is invoked, using the START=EMER keyword, System Initialization:

- Repositions the system log.
- Cold-starts the APT, PCT, TCT, FCT, CSA, Transient Data map, and Temporary Storage map.
- Links to the Recovery Utility program, which reads the system log and builds recovery data and tables that are written to the Restart Data Set.
- Links to the Transaction Backout program, which reads the recovery data and backs out the effects of transactions that were in process immediately before termination.

System Initialization builds the CICS/VS nucleus, initializes data sets, opens system and user data sets, constructs and initializes tables, and builds the CICS/VS dynamic storage pool.

The CICS/VS Program Library is accessed by DOS/VS LOADs, or by using BSAM on OS, to build the CICS/VS nucleus, load tables, load resident application programs, and initialize the Processing Program Table (PPT).

Communication with CICS/VS nucleus modules is required during post-initialization processing, both by System Initialization and by application programs running at this time. System Initialization always interfaces with Storage Management, Task Management, Time Management, and Program Management and may interface with Temporary Storage Management, Transient Data Management, File Management, and System Recovery. All communication with CICS/VS nucleus modules is performed under Terminal Management's Task Control Area (TCA), which is borrowed temporarily as a communication vehicle.

System Initialization passes control to Terminal Management (entry point DFHTCP). In DOS systems, System Initialization remains in the first 12K of the virtual storage partition. This area contains the common routines used by CICS/VS management modules:

- BLDL routine
- Program Loader subtask
- Write-to-Operator subroutine

SYSTEM TERMINATION

The Master Terminal program transfers control to the System Termination (DFHSTP) using a Program Management DFHPC TYPE=XCTL macro instruction when a CSMT SHUTDOWN request has been entered by the CICS/VS master terminal operator.

The System Termination program provides for the orderly shutdown of CICS/VS operation. Terminals are allowed to quiesce, the console operator is notified that termination is in process, data sets are closed, and system statistics are extracted. In addition, the system termination program records vital information about CICS/VS so that the system initialization program can warm-start the system.

The System Termination program operates in stages, as follows:

Stage 1:

During the first quiesce stage of system termination, control is transferred to the first of a series of programs defined in a Program List Table. Each program in the first portion of the table is given control in the sequence of its occurrence in the table. During this stage, all system activity is quiesced except those transactions specified in a transaction list table. Such transactions remain eligible for execution and have access to all CICS/VS services, including terminal control. Transactions not in the specified table are not initiated.

Stage 2:

When execution of the programs specified in the first portion of the PLT is completed, the second quiesce stage begins. During this stage, all terminal activity is quiesced and control is transfered to the first program in the second portion of the user defined program list table. Each program is given control in the sequence of its occurrence in the table and has access to all CICS/VS functions except terminal control.

Stage 3:

When the last program in the specified PLT has completed execution, system termination closes all data sets if the system is DOS, records system statistics, records warm-start information, and returns control to the operating system.

The Transaction List Table (XLT) and PLT are loaded (by Program Management) from the CICS/VS Program Library.

Terminal activity is quiesced by setting an indicator in the CSA. This tells Terminal Management not to attach any transactions other than those specified in the XLT. The termination task logically disconnects itself from the physical terminal to allow other activity on that terminal.

The termination task allows all other tasks (except for any journal tasks) to complete before linking to the first program specified in the first portion of the PLT.

When all programs in the first portion of the PLT have executed, terminal activity is quiesced completely by setting another indicator in the CSA. The ICE, AID, and BCA chains are broken (addresses saved in the TWA), and the programs specified in the second portion of the PLT are executed.

CICS/VS-DL/I Interface and Journal Management are terminated; Temporary Storage Management is requested to output its buffer; statistics are taken by a link to the System Statistics program; on a DOS system all files are closed; and a keypoint is taken by the Keypcint program.

Control is returned to the operating system, with or without a dump (depending upon the parameters specified in the shutdown request causing termination).

If an immediate shutdown is requested, tables are not loaded, terminals are not quiesced, and the programs specified in the PLT are not executed.

HIGH-LEVEL LANGUAGE PREPROCESSOR

The high-level language preprocessor program (DFHPRPR) prepares a high-level language program for input to the Assembler. The assembler then generates the high-level language statements for CICS/VS macro instructions for input to the high-level language compiler.

The input and output data sets are opened and the source records read. The first 16 columns of each record are scanned for "DFH" or "CALL". If this scan fails then the source record is written out with a REPRO card preceding it. DL/I calls are changed to CICS/VS calls; CICS macro instructions are left unchanged except for some conversions from hexadecimal to decimal and some systematic changes of the prefix DFH. An END statement is appended to the output.

COMMAND LANGUAGE TRANSLATOR

The Command language translator is a utility program (DFHEXP) which runs off-line to translate CICS/VS application programs using the command level interface. It converts the EXEC statements into CALL statements in the language in which the EXEC CICS statements are embedded.

There are four versions:

- OS COBOL
- OS PL/I
- DOS COBOL
- DOS PL/I

The translator manages storage by creating a stack from a single area allocated at the start of the program.

Since the input is free format the translator moves input into a buffer area that can hold input spanning two or more input records. The analysis of the source is table driven. The statements are first parsed at the highest level; that is, constructions of the form:

are recognized. Keywords and options are then recognized and the keywords and parameters converted to a string of bits. Duplicate keywords are detected during this scan.

The replacement string for each EXEC CICS command is built up in another string. In the case of PL/I, the replacement field will contain a single CALL statement. In the case of COBOL, the string will contain a series of MOVE statements followed by a CALL statement.

Errors in the source may be detected. Spelling corrections are made to the source, and then unrecognizable or duplicate keywords and options are ignored. The preprocessor produces error diagnostics which appear on the output listing.

SYSTEM LOG/JOURNAL UTILITIES

These utilities are used to preformat magnetic tapes or disk extents to be used as system logs or journals. They also allow the user to place an end-of-file mark on magnetic tapes to be used as journals, following an abnormal system termination.

FORMAT TAPE

To prevent invalid recovery due to erroneous data on the System Log, all tape volumes used for this purpose should be pre-formatted using the format tape program (DFHFTAP). Formatting magnetic tapes facilitates finding the end of file if the system terminates abnormally without writing an end-of-file mark. This function can be performed by the stand-alone program DFHFTAP, which provides the following services:

- Opens message data set
- Acquires 32K of storage for a record area
- Opens tape volume and writes binary zero records

If any of these services cannot be performed successfully, a message is returned and the program is terminated. Otherwise, binary zeros are written until the end-of-file condition is encountered or an I/O error occurs. If an I/O error occurs, no recovery of the write error is performed and no more formatting occurs. The volume is closed and another volume is requested. If no other volume is to be formatted, the program is terminated. If end of file occurs (normal end), the volume is closed and another volume is requested. If no other volume is to be formatted, the program is terminated; otherwise, processing continues with opening and writing of records until all volumes have been formatted.

TAPE END OF FILE

The tape end-of-file program (DFHTEOF) is run as a standalone program or attached by the System Initialization program (DFHSIP) during the initialization phase of an emergency restart. It performs the following functions:

- Verification of tape volumes
- Verification of log records collected as part of CICS/VS run before system failure
- Writing end of file

The log volume is opened and verified. If an incorrect volume is mounted, volume swapping takes place until either the correct volume is mounted or swapping is discontinued without finding the correct volume. In the latter case, the program is terminated.

As the file is processed, the label records of blocks on the file are checked to verify thay they are in ascending sequence. Verification of these label records is performed as follows:

- Creation date equal to or greater than that specified on the volume label.
- Volume sequence number equal to or greater than that specified on the volume label.
- Run start time equal to that specified on the volume label.

The end of valid log data is assumed under either of two conditions:

- Verification fails during validation of label records.
- Two consecutive I/O errors are encountered.

If either of these conditions occurs, the tape volume is backspaced over the appropriate number of records and an end-of-file record is written.

DUMP UTILITY

The output from Dump Management, generated during the execution of CICS/VS, is formatted and printed by the dump utility program (DFHDUP). This program operates in batch mode while one of the dump data sets is closed. Each area, program, and table entry is identified, formatted, and printed separately, with both actual and relative addresses to facilitate analysis. The user can select single or double spacing of dumps when the dump utility program is executed.

TRANSACTION BACKOUT PROGRAM

The Transaction Backout program (DFHTBP) is invoked by System Initialization in the event of an emergency restart. It reads backout data from the restart data set, previously written there by the Recovery Utility program.

The Transaction Backout program reads data from the restart data set. This data comprises the task, message, DL/I, and file backout tables, and transaction backout data.

The Transaction Backout program saves messages for in-flight tasks and unresponded-to output messages in a Temporary Storage user message cache. Each message is stored under the identification DFHMxxxx, where xxxx is the terminal identification, of the terminal to which it belongs

and is available to the user. Output messages for which no positive response was found are also placed in Temporary Storage whence Terminal Management can fetch them for re-presentation.

Data is read using the Keypoint program; Storage Management is invoked to acquire and free storage areas; user-written exits, if present, are given control; and Transient Data Management is used to write messages, which also go to the operator console.

FORMATTED DUMP

The Formatted Dump Program will be invoked by abnormal termination of CICS/VS, by specification of DUMP at CICS shut-down time, or by the issue of a CSMT SNAP command by the Master Terminal Operator.

The output of the Formatted Dump Program can be in one of three forms, depending on the specification made either at CICS start-up using one of the start parameters, or at system initialization by the FDP parameter of the DFHSIT macro. The three forms of output thus specified are:

1. A dump of the supervisor and CICS partition [DUMP],

2. A dump of the CICS partition only [PDUMP], or

3. A dump of the CICS partition [PDUMP] followed by a series of control blocks, each dumped in as logical an order as possible. Fields to be highlighted in each control block will follow the hexadecimal dump of the appropriate control block.

The Formatted Dump Program (DFHFDP) consists of three modules: DFHFDA, DFHFDB, and DFHFDC.

DFHFDA is the control module of the FDP, and acts as an interface module to CICS/VS and the operating system. It has only one entry point which is used for taking a dump with or without formatting. It contains all system-dependent code and output routines. It also contains the code which is to be executed at CICS initialization time. According to the option selected at CICS start-up time, DFHFDA will issue DUMP, or issue PDUMP and return, or issue PDUMP, call DFHFDB and return.

The main routines and subroutines of DFHFDA are:

- Initialization (at CICS/VS start-up)
- Main dump routine
- Sub-routines used by DFHFDB
- First level program check handler
- Second level program check handler
- Initial working storage area
- Communication area for DFHFDP (DFHFDPDS)

DFHFDB is the module which performs the bulk of the work in producing the formatted dump. It consists mainly of an interpreter which "executes" the text contained in DFHFDC. It calls DFHFDA to perform all operating system-dependent functions, and to perform the actual formatting of the output. The functions of its main routines are:

- Initialization - reinitializes the necessary areas of the interpreter so that the program is serially reusable.
- Primes the interpreter by starting it with a pointer to the Common System Area (CSA) and the text string for the CSA, which must be the first string in DFHFDC.
- Queue scan routine - this is the "scheduler" of the interpreter; it decides which control block should be processed next.
- Work element preparation - having decided which control block is to be processed next, generates the necessary pointers and control information.
- IFETCH - fetches the operation code of the next descriptor and branches to the appropriate descriptor processing routine.
- Descriptor processing routines
- Termination processing routines
- Error processing routines

Module DFHFDC contains the text which is interpreted by DFHFDB, and consists of two CSECTs. The first of these contains fixed length entries, one for each type of control block to be dumped, and acts as an index to the second CSECT which contains the text strings, one for each control block type. The text strings contain "instructions" which describe to the interpreter where it will find pointers to other control blocks, which fields should be formatted, etc. All lengths and offsets used in DFHFDC are determined from the appropriate DSECT, so that any change of position or length in a DSECT will be effected in the Formatted Dump Program by reassembly of DFHFDC only. A change of type, however, from character to hexadecimal for example, would require a source change to DFHFDC.

Chapter 11. Application Services

CICS/VS provides several functions designed to perform services closely associated with user applications. These services rely on CICS/VS system management functions to achieve their objectives and can be considered as logical extensions to the user-written application programs. The application services are:

- Basic mapping support
- Data Interchange Program
- 2260 compatibility
- Command Interface
- Built-in functions

BASIC MAPPING SUPPORT

The Basic Mapping Support (BMS) function allows the CICS/VS application programmer to have access to input and output data streams without including device-dependent code in the CICS/VS application program.

Maps are assembled offline using CICS/VS macro instructions. The user defines and names fields and groups of fields that can be written to and read from the devices supported by BMS. The assembled maps contain all the device-dependent control characters necessary for the proper manipulation of the data stream.

Associated with each map is a table of field names that is copied into each application program that uses the map. Data is passed to and from the application program under these field names. The application program is written to manipulate the data under the various field names so that alteration of a map format does not necessarily lead to changes in program logic. New fields can be added to a map format without making it necessary to reprogram existing applications.

Output data may be supplied from the application program by placing the data in the table under the appropriate field name. As an alternative, output maps can contain field default data that is sent when data supplied by an application program is not present. This facility permits the specification of titles, headers, etc., for output maps.

Optionally, displaying all the default data can be suppressed by the application program for any output map. Each time a map is used, the application program can temporarily modify the attributes of any named field in the output map. Output map fields with no field names can contain default data, but the application program cannot replace the default data or modify the attributes of unnamed fields.

For input, the user assembles a map defining the fields that can be written to and received from a particular device. Any data received for a particular field is moved across using the field name in the symbolic storage definition for the map. Pen-detectable fields defined in an input map are flagged as detected if present in a 3270 input stream. An input map for a particular case can specify a subset of the fields

potentially receivable; any fields received and not represented in that map are discarded. This permits the number of keyable and selectable fields from a map to be changed without making it necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS/VS program load library or, in the case of assembler language, can be coded in the application program. When a map stored in the program load library is referenced by BMS, a copy is automatically retrieved by CICS/VS without application program action. Multiple users of a map contained in the program load library share a single copy in main storage.

BMS permits any valid combination of field attributes to be specified by the user when generating maps. Inclusion of BMS in CICS/VS is a system generation option and its inclusion does not prevent the application program from accessing a particular device in native mode (without using BMS). Intermixing BMS and native mode support for a terminal from the same application program may yield unpredictable results. When using mixed mode support, it is the user's responsibility to ensure correct construction and interpretation of native mode data streams.

BMS permits the application program to pass a native mode data stream (that has already been read in, and provided the screen has been formatted if a 3270 is being used) and to interpret this data stream according to a given input map. This facility allows data entered with the initial reading of a transaction to be successfully mapped using BMS.

Basic Mapping Support provides the following services:

- Message routing
- Terminal paging
- Device independence

MESSAGE ROUTING

This service permits the application program to send an output message to one or more terminals not in direct control of the transaction. The message is automatically sent to a terminal if the terminal status allows reception of the message. If a terminal is not immediately eligible to receive the message, the message is preserved for that terminal until a change in terminal status allows it to be sent. The message routing function is used by the CICS/VS message-switching transaction.

TERMINAL PAGING

This feature allows the user to prepare more output than can be conveniently or physically displayed at the receiving terminal. The output can then be retrieved by pages in any order; that is, in the order they were prepared or by skipping forward or backward in the output pages.

Terminal paging also provides the ability to combine several small areas into one area, which is then sent to the terminal. This enables

the user to prepare his output without regard for the record size imposed by the output terminal.

CICS/VS provides the terminal operator with a generalized page retrieval facility that can be used to retrieve and dispose of pages.

DEVICE INDEPENDENCE

This feature allows the user to prepare his output without regard for the control characters required for message heading, line separation, etc. Input to device independence consists of a data string with optional new-line characters.

Device independence divides the data string into lines no longer than those defined for the particular terminal. If new-line characters appear occasionally in the data string to further define line lengths, they are not ignored. CICS/VS inserts the appropriate leading characters, carriage returns, and idle characters and eliminates trailing blanks from each line.

CICS/VS allows the user to set horizontal and vertical tabs on those devices which support the feature (for example, 3767 and 3770). For such devices, CICS/VS supports data compression inbound and data compression outbound, based on the tab characteristics in the data stream under control of the appropriate maps.

BMS MODULES

Basic Mapping Support consists of a number of modules, each of which has interfaces with other BMS modules, CICS/VS management components, and application programs. The maps that are handled by CICS/VS BMS may be new maps, created to use CICS/VS BMS mapping capabilities, or old maps, created for pre-VS BMS.

Pre-VS BMS Mapping Module

The pre-VS BMS Mapping Module (DFHBMSMM) is called in response to requests for BMS services code in the form established for versions of CICS released before CICS/VS.

A pre-CICS/VS DFHBMS TYPE=IN, MAP, or OUT macro request by an application program communicating with a 3270 terminal passes information, in the TCA, through Program Management to DFHBMSMM.

A CICS/VS DFHBMS TYPE=IN, MAP, or OUT macro instruction using pre-CICS/VS maps and DSECTs to communicate with a 3270 terminal passes information, in the TCA, through the Mapping Control program to DFHBMSMM. Maps are either passed by the application program or loaded by DFHBMSMM.

The address of a Terminal Input/Output Area (TIOA) is supplied by the application program for TYPE=MAP or TYPE=OUT requests and by Terminal Management for TYPE=IN. Terminal Management is used to read the data for a TYPE=IN request.

DFHBMSMM communicates with Storage Management to obtain and release work areas and buffers for mapping operations, and communicates with

Program Management to load and delete maps required for mapping operations.

Data Stream Build

The Data Stream Build program (DFHDSB) addresses the page buffer, which was composed by the Page Build program. The page buffer contains lines of output data that are to be written to a terminal other than a 3270. The following functions are performed by the Data Stream Build program on the data in the page buffer:

- Truncates trailing blanks within data lines.
- Substitutes strings of physical device control characters for logical new-line characters that terminate each line of data.
- Provides a function management header (FMH) for some VTAM-supported devices.
- Allows horizontal and/or vertical tab processing.

DFHDSB is entered from the Page Build program to process the page buffer. For TYPE=NOEDIT, page buffer compression is skipped and control is given to the Terminal Page Processor. If not TYPE=EDIT, the appropriate device control characters for the target device are selected for substitution. After compression of the page buffer data, DFHTPP is called to provide disposition of the page.

Non-3270 Input Mapping Program

The Non-3270 Input Mapping program (DFHIIP) is called in response to requests for BMS services involving terminals other than 3270 devices.

A DFHBMS TYPE=IN or TYPE=MAP request by an application program communicating with other than a 3270 terminal passes information, in the TCA, through the Mapping Control program to DFHIIP.

The map required for an operation is either passed by the application program or loaded by DFHMCP.

The address of a Terminal Input/Output Area (TIOA) is supplied by the application program for TYPE=MAP and by Terminal Management for TYPE=IN.

Normally Terminal Management is used to read the data for a TYPE=IN request. DFHIIP calls Storage Management to obtain and release buffers for mapping operations. For the Batch logical unit, if INBFMH=DIP is specified in the PCT entry for the transcation code, the Data Interchange program is used to read the data.

Mapping Control Program

The Mapping Control program (DFHMCP) is the interface between application programs and the modules which perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary storage areas, or the application program.

This program is entered when an application program issues a DFHBMS request for Basic Mapping Support services. It may also be called by Task Management to process a Deferred Work Element (DWE) if an application program terminates and there are partial pages in storage or

the Message Control Record (MCR) created during execution of the task has not been placed in temporary storage.

The expansion of the DFHBMS macro instruction and the application program insert data into fields in the TCA. The following information is returned to the requestor in fields of the TCA: error codes, page overflow information, and a list of completed pages (if TYPE=RETURN was specified in the request).

A Terminal Management DFHTC TYPE=SAVE macro instruction is issued if TYPE=SAVE was specified in the DFHBMS macro instruction.

The Mapping Control program communicates with Temporary Storage Management to output the MCR for routed or stored messages (TYPE=ROUTE and/or TYPE=STORE was specified). A DFHTS TYPE=PURGE macro instruction is issued to request that a message be purged from temporary storage if a DFHBMS TYPE=PURGE request is issued.

The Mapping Control program communicates with Storage Management to:

- Acquire and free storage in which the MCR is built (TYPE=PAGEOUT after TYPE=STORE and/or TYPE=ROUTE)
- Acquire and free storage in which to copy the message title (TYPE=ROUTE,TITLE=symbolic address or YES)
- Acquire storage to build Automatic Initiate Descriptors (AIDs) for non-routed message or routed messages to be delivered immediately (TYPE=PAGEOUT)
- Acquire a BMS work area (OSPWA) at the time of the initial BMS request
- Acquire and free an area used for user request data if a TYPE=PAGEOUT must be simulated before processing the user's request
- Free the returned page list (TYPE=PURGE)
- Free map copies if TYPE=PAGEOUT and pages were being built in response to TYPE=PAGEBLD requests
- Free Terminal Type Parameters (TTPs) (TYPE=PAGEOUT)

The Mapping Control program communicates with Program Management to:

- Load and delete map sets
- Link to the Page Retrieval program to process one or more pages of a message if TYPE=PAGEOUT and CTRL=RETAIN or CTRL=RELEASE
- Abnormally terminate a task if uncorrectable errors occur
- Link to the BMS Mapping Module (DFHBMSMM) if a pre CICS/VS map is loaded

The Mapping Control program communicates with Time Management to:

- Initiate transaction CSPQ
- Obtain the current time of day, which is then used to time-stamp AIDs for routed messages

- Initiate transaction CSPS for messages to be delivered at some future time

The Mapping Control program communicates with Task Management to schedule transaction CSPG for every terminal that is to receive a routed message to be delivered immediately. Transient Data Management is used to send error and informational messages to the master terminal.

Route List Resolution is used to collect terminals from a user-supplied route list or from the entire TCT by terminal type, and build a Terminal Type Parameter (TTP), which controls message building, for each terminal type. It is also used to build a one-element TTP for the originating terminal.

3270 Mapping

The 3270 Mapping program (DFHM32) is called in response to requests for BMS services involving terminals of the 3270 Information Display System, 3650 -3275 Host conversational system, and 3790 - 3270 Emulator.

A DFHBMS TYPE=PAGEBLD, TEXTBLD, OUT, STORE, or RETURN macro request by an application program communicating with one of these terminals passes information, in the TCA, through the Mapping Control program and the Page and Text Build program to the 3270 Mapping program.

A DFHBMS TYPE=IN or TYPE=MAP macro request by an application program communicating with a 3270 terminal passes information, in the TCA, through the Mapping Control program to The 3270 Mapping program.

Maps are either passed by the application program or loaded by the Mapping Control program.

The address of a Terminal Input/Output Area (TIOA) is supplied by Terminal Management for TYPE=IN requests and by the application program for all other requests. Terminal Management is also used to read the data for a TYPE=IN request.

The 3270 Mapping program communicates with Storage Management to obtain and release buffers for mapping operations. All output requests are sent to a designated destination by the Terminal Page Processor.

Page and Text Build

The Page and Text Build program (DFHPBP) processes all BMS output requests (DFHBMS TYPE=OUT, STORE, RETURN, or PAGEOUT). It performs the following functions:

- positions the data in the page, either by actually placing it in a buffer or by copying it and adjusting the map for a 3270 (TYPE=PAGEBLD)
- places the data into the page buffer or, for the 3270, copies it and generates a map (TYPE=TEXTBLD)
- inserts device dependencies for other than 3270 Information Display System devices

The Page and Text Build program is entered from the Mapping Control program to process all BMS output requests. It is called once for each Terminal Type Parameter (TTP) on the TTP chain. The Page and Text Build program returns control to the Mapping Control program when request

processing is complete, or when the page must be written out before a TYPE=PAGEBLD request can be processed and an OFLOW=symbolic address operand was specified.

For a TYPE=PAGEBLD request for a 3270, the map is copied and chained to the TTP. For a TYPE=TEXTBLD request for a 3270, a dummy map is created and chained to the TTP. When a page is complete, control is given to 3270 Mapping (The 3270 Mapping program), which combines the map copies chained to the TTP and maps the data.

The Page and Text Build program communicates with Storage Management to:

- Acquire and free buffers in which pages are built
- Acquire storage for a copy of the user's data and map for TYPE=TEXTBLD or TYPE=PAGEBLD

  For 3270s the user's data is copied only if routing is being used or if SAVE was specified. If the data is copied then one GETMAIN is used for both map and data.

The Page and Text Build program requests Program Management to abnormally terminate a transaction (DFHPC TYPE=ABEND) if certain uncorrectable errors occur.

A TYPE=TEXTBLD request for a 3270 causes a map set consisting of one dummy map to be passed to 3270 Mapping (The 3270 Mapping program). The map has one field with attributes FREEKB and FRESET. If the page is being constructed for a 3270, control is given to The 3270 Mapping program to map the data and then to DFHTPP to output the page. Otherwise, device dependencies are inserted in the page and control is given to the Terminal Page Processor to output the page.

Route List Resolution Program

The Route List Resolution program (DFHRLR) builds Terminal Type Parameters (TTPs), which are the main blocks for building and outputting data in BMS.

The Route List Resolution program is called by the Mapping Control program to determine the grouping of terminal destinations. If data is to be routed, DFHRLR groups the terminals in the user's route list by terminal type and builds a routing TTP for each type. The address of the first routing TTP in the chain of TTPs is placed in OSPTTP. If data is not to be routed, a direct TTP is built for the originating terminal and its address is placed in OSPDTTP.

The Route List Resolution program communicates with Storage Management to acquire storage for the TTP. Program Management services are requested by issuing a DFHPC TYPE=ABEND macro instruction if certain uncorrectable errors occur.

Terminal Page Processor

The Terminal Page Processor (DFHTPP) puts completed pages to a destination specified in the BMS output request (TYPE=CUT sends to the originating terminal; TYPE=STORE directs to Temporary Storage; and TYPE=RETURN directs to a list of completed pages that are returned tc the application program).

The Terminal Page Processor is entered from 3270 Mapping for 3270s, from FASTER 2260 Compatibility for FASTER 2260 compatibility output, and from Data Stream Build for other devices.

The Terminal Page Processor communicates with Storage Management to obtain:

- The return list (to store the address of completed pages to be returned to the programmer)
- Deferred Work Elements (DWEs), which ensure that message control information is written to disk even if the programmer neglects to issue a DFHBMS TYPE=PAGEOUT request
- Storage for a list that correlates pages on temporary storage with the logical device codes for which they are destined.

Temporary Storage Management is used to store pages and the Message Control Record (MCR) for messages stored on Temporary Storage. The Terminal Type Parameter (TTP) controls the formatting of a message for a particular terminal type, for example, 2741.

Terminal Page Cleanup Program

The Terminal Page Cleanup program (DFHTPQ) checks the chain of Automatic Initiate Descriptors (AIDs) to detect and delete AIDs that have been on the chain for longer than the purge-delay time specified at system generation (DFHSG PROGRAM=BMS,PRGDLAY=hhmm).

The Terminal Page Cleanup program is initiated the first time by the Mapping Control program using Time Management. Thereafter, it reinitiates itself.

The program also calls Storage Management to free AIDs which have been purged and to acquire storage for notification messages. Transient Data Management is used to send notification messages. Time Management is used to obtain the current time and to reinitiate this task. The Terminal Page Cleanup program communicates with Temporary Storage Management to retrieve and replace Message Control Records (MCRs) and to purge messages.

Page Retrieval Program

The Page Retrieval program (DFHTPR) processes messages built by BMS and placed in Temporary Storage.

The Page Retrieval program is entered from Program Management to do one of the following:

- Display the first page of a routed message
- Display subsequent pages of a message at a terminal for which TYPE=PAGEOUT,CTRL=AUTOPAGE was specified
- Process paging commands from a terminal
- Process transaction CSPG when it is entered at the terminal
- Purge a message displayed at the terminal if the terminal is in display status and other than a paging command is entered at the terminal

The Page Retrieval program is entered from the BMS Mapping Control program to display the first page of a message originated at the terminal. If CTRL=RETAIN was specified in the BMS request, the Page Retrieval program reads from the terminal and processes paging commands until other than a paging command is entered.

The Page Retrieval program communicates with Storage Management to:

- Acquire and free Message Control Blocks (MCBs)
- Free terminal page storage and Message Control Record (MCR) storage
- Acquire storage for informational and error messages to be sent to the destination terminal and the master terminal
- Free an Automatic Initiate Descriptor (AID) taken off the AID chain
- Acquire and free storage for a route list constructed in response to a COPY command entered at a terminal
- Acquire a TIOA into which to place a device-independent page when performing the COPY function

Temporary Storage Management is used to retrieve and replace MCRs and to retrieve and purge pages. Basic Mapping Support is used to display error and informational messages at a requesting terminal and to send a page to the destination terminal in the COPY function. Task Management is used to retain exclusive control of a MCR while it is being updated.

The Page Retrieval program communicates with Time Management during error processing when a Temporary Storage identification error is returned while attempting to retrieve a MCR. Up to four retries (each consisting of a one-second wait followed by another attempt to read the MCR) are performed. (The error may be due to the fact that an MCR has been temporarily released because another task is updating it. If so, the situation may correct itself, in which case a retry is successful.)

Terminal Management is used to read in the next portion of terminal input after a page or informational message is sent to the terminal when TYPE=PAGEOUT,CTRL=RETAIN was specified. Transient Data Management is used to send error or informational messages to the master terminal.

The Terminal Output macro instruction (DFHTOM) is issued to provide an open subroutine that puts a completed page out to the terminal.

Terminal Page Scheduling Program

The Terminal Page Scheduling program (DFHTPS) processes messages that have been scheduled for delayed delivery. An Automatic Initiate Descriptor (AID) is built and scheduled for every terminal specified in the Message Control Record (MCR) that Time Management provides.

The Terminal Page Scheduling program is called by Program Management at the time when a delayed message is to be sent. The program also calls Time Management to obtain the Message Control Record (MCR) that has the IDs of all terminals that are to receive the message. The Terminal Page Scheduling program communicates with Temporary Storage Management to replace the MCR, since Time Management released the MCR while retrieving it.

Time Management has created an Interval Control Element (ICE) for the time-dependent request that is now to be serviced. The Terminal Page

Scheduling program refers to many fields of the MCR (supplied using Time Management) when building the AIDs required for the message.

The Terminal Page Scheduling program communicates with Storage Management to acquire storage for the AIDs that it constructs. After the AIDs have been constructed, The Terminal Page Scheduling program communicates with Task Management to schedule the AIDs for processing.

DATA INTERCHANGE PROGRAM

The Data Interchange program (DFHDIP) is invoked by the DFHDI CICS/VS macro, and is also invoked by BMS when dealing with a batch LU. For a RECEIVE operation the program will get a message from the Batch Logical Unit and, after removing any FMHs, pass the message and destination name to the application program. For all other requests (ADD, ERASE, REPLACE etc.) appropriate FMHs are built and transmitted to the Batch Logical Unit together with the user's message, if one is specified.

Errors (other than set-up errors) are signalled to the user by return codes. Set-up errors cause a CICS/VS abnormal termination.

2260 COMPATIBILITY

Under BTAM, the user can run his currently operational 2260-based transactions from a 3270 Information Display System. This facility is known as 2260 compatibility.

During CICS/VS system generation, the user must request that 2260 compatibility be included, thereby generating the necessary code to provide conversion of 2260 data streams from user-written application programs to the appropriate 3270 data stream format. When the 3270 operates with a compatibility transaction, incoming data from the 3270 is converted and presented to the user-written application program in 2260 format. In most cases, no changes are required to the user-written program.

Because 2260 compatibility is specified by transaction as well as by terminal, non-2260-based transactions have full access to all facilities of the 3270.

The FASTER 2260 compatibility routines provide 2260 compatibility on a 3270 for users of the FASTER Language Facility (FLF). However, 2260 compatibility is not available for 3270 support through VTAM.

DFHFIP is invoked by Terminal Management after each successful Read operation on a 3270 terminal that was defined as being FASTER 2260 compatible. All input that is not in 3270 native mode (no SBA as a first character) is rearranged to provide to the application program only the data between the start-of-message indicator (SMI), if present, and the cursor. If a new-line character (NL) is encountered, the remainder of the 2260 logical line is dropped.

DFHF2P is invoked by the BMS program for each page of output that is not in 3270 native mode and is destined for a 3270 that was defined as being FASTER 2260 compatible. The program arranges the output page to appear as it would on a 2260 by inserting user-defined SMI and NL characters in the data stream.

EXEC INTERFACE PROGRAM

When Program Management loads an application program that has been translated by the command language translator, the EXEC interface program is invoked. This initializes the command-level interface environment which involves setting up up the EXEC interface structure (which is used for housekeeping), and the EXEC interface block(which is used by the application program). The EXEC interface program then invokes the application program.

When the application program reaches the expansion of an EXEC CICS command the EXEC interface program will be called, passing parameters according to the keywords and options on the EXEC CICS command. The EXEC interface program branches to separate routines according to the operation used. These routines contain calls to the CICS/VS modules.

When the response from CICS/VS is other than the normal expected response an exceptional-condition handler gains control. This routine provides the linkage to the locations set up in the EXEC CICS CHECK command.

Finally the Interface program returns to the application program.

BUILT-IN FUNCTIONS

Several commonly used functions are available to the application programmer through the use of CICS/VS macro instructions. These are functions that generally were coded as separate subroutines by the programmer. These capabilities, referred to as built-in functions, are as follows:

- Table search
- Phonetic conversion
- Field verify/edit
- Bit manipulation
- Input formatting
- Weighted retrieval

The Built-In Functions program (DFHBFP) may be generated with either (or both) of two options: (1) the basic set, which includes Table Search, Phonetic Conversion, Field Edit, Field Verify, Bit Manipulation, and Input Formatting; and (2) Weighted Retrieval. If generated, any of these functions can be called by any application program.

When the Built-in Function are used in an application program, the symbolic storage definition for the communication area of the Built-In Functions program must by copied into the common control communication area of the application program communication section of the program's TCA. This copying is achieved by issuing a DFHBFTCA macro instruction, which must immediately follow the statement that copies the TCA and the user's definition of a TWA, if any, in the application program.

Built-in Functions are requested by DFHBIF macro instructions, which establish fields in the requesting program's TCA for communication with the Built-in Function program.

TABLE SEARCH

The table search built-in function provides the application program with the means of conveniently searching a table for a specific entry and having some value within that entry returned. The user can elect to have a default value returned if the desired entry is not in the table.

PHONETIC CONVERSION

The phonetic conversion built-in function provides the CICS/VS user with the capability of converting a name into a partial key, which can then be used to access a data base name file. The key produced is based upon the sound of the name. This means that names that sound similar but are spelled differently, generally produce like keys. For example, the names SMITH, SMYTH, and SMYTHE produce a phonetic key of S530. Likewise, the names ANDERSON, ANDRESEN, and ANDRESENN produce a phonetic key of A536. A CICS/VS subroutine to convert keys in a similar manner is provided for use by offline programs. Together, these facilities allow the CICS/VS user to organize files of names so that they can be accessed by names that may be misspelled, mispronounced, or misunderstood.

FIELD VERIFY/EDIT

The field verify function enables CICS/VS application programmers to verify the contents of a data field as either alphabetic or numeric and branch to the appropriate routine. Any field can be checked for the following:

1. Entirely alphabetic: blanks or A-Z
2. Entirely EBCDIC digits: 0-9
3. Entirely packed decimal (COMPUTATIONAL-3 in ANS COBOL or FIXED DECIMAL in PL/I).

The field edit function allows the application program to pass a field containing EBCDIC numbers intermixed with other values and receive a result with all non-numeric characters removed. The result can be in EBCDIC format.

BIT MANIPULATION

The bit manipulation function allows the high-level language program to set or test the value of one or more bits in a byte and to branch according to the result.

INPUT FORMATTING

This built-in function allows the application program to convert free-format input from the terminal operator into a predefined fixed format that is more easily manipulated. The free-format input can be positional or keyword oriented. If positional, the data must be keyed in a specific sequence; for example, last name, first name, middle initial.

If the terminal input is keyword oriented, the application program must define a set of symbolic keywords that identify the data to be entered. In addition, the user installation must define a keyword-prefix character and a field-separator character.

The symbolic keywords are defined within the application program, providing a high degree of dynamic flexibility when requesting input from the terminal. The keyword-prefix character and the field-separator character are defined for the entire system.

WEIGHTED RETRIEVAL

The weighted retrieval function allows the user to search a specified group of records on a VSAM data set (defined in the File Control Table), selecting only those records that are closest to the selection criteria he provides. Selection is made on the basis of a qualification weight developed for each record in the group of records being searched. Records are presented to the user in order of decreasing weight. Selection criteria can be fixed for a given transaction type, can depend upon variables entered from the terminal as a part of the transaction, or can include both fixed and variable factors.

The weighted retrieval function is supported only for VSAM data sets defined in the File Control Table. The Weighted Retrieval built-in function communicates with Storage Management and File Management.

Index

Customer Information Control System/Virtual Storage (CICS/VS)
Introduction to Program Logic

READER'S COMMENT FORM

SC33-0067-0

Your comments about this publication will help us to improve it. Please give specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

Number of your latest Technical Newsletter for this publication .....

CUT ALONG DOTTED LINE

Reply requested:
Yes ☐
No ☐

Name .....
Job Title: .....
Company: .....
Address: .....
.....
.....
..... Zip .....

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SC33-0067-0

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 813 HP
1133 Westchester Avenue
White Plains, New York 10604

Fold Fold

CICS/VS Introduction to Program Logic Printed in U.S.A. SC33-0067-0

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

SC33-0067-0

CICS/VS Introduction to Program Logic Printed in U.S.A. SC33-0067-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)