IBM

Customer
Information
Control
System
CICS/MVS

Licensed Program
Version 2.1

Program Number
5665-403

Diagnosis Reference

LC33-0517-0

First Edition (December 1987)

This edition applies to Version 2 Release 1 (Version 2.1) of the licensed program Customer Information Control System/MVS (CICS/MVS), program number 5665-403.

This edition is based on the earlier book for CICS/OS/VS 1.7, LC33-0243-0 (which remains current and applicable for users of Version 1.7). Technical changes are indicated by vertical lines to the left of the changes.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below. Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

International Business Machines Corporation, Department 6R1H,
180 Kost Road, Mechanicsburg, PA 17055, U.S.A.

or to:

IBM United Kingdom Laboratories Limited, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

What This Book Is About

This book, in conjunction with the CICS source program listings, helps you to understand the inner workings of CICS. The information in this book should help you identify the causes of a failure in a CICS program.

Who This Book Is For

This book is for system programmers and IBM Service personnel who maintain CICS in an MVS/XA environment.

What You Need to Know to Understand This Book

This book assumes system programming experience, and experience of computer applications. It also assumes that you have a basic knowledge of CICS. See the *CICS/MVS Facilities and Planning Guide*, with particular reference to the chapter on CICS operations, which gives a summary of the overall structure of CICS and general information on how CICS processes applications and transactions.

How to Use This Book

The four parts of this book are structured hierarchically, beginning with high-level introductory information and progressing through lower-level design and organization parts, and ending with a directory of source modules. If you are new to CICS, you will probably need to begin by reading the high-level introductory information. If you are not new to CICS, you may prefer to start with one of the lower-level parts.

Notes on Terminology

In this publication, the term **VTAM** refers to ACF/VTAM, and to the Record Interface of ACF/TCAM. The term **TCAM** refers to both TCAM and the DCB interface of ACF/TCAM. The term **CICS** refers to CICS/MVS.

Book Structure

Bibliography

CICS/MVS Version 2 Release 1 Library

Evaluation and Planning

- Brochure GC33-0503
- General Information GC33-0155
- Facilities and Planning Guide SC33-0504
- Release Guide GC33-0505

General

- Library Guide GC33-0356
- Master Index SC33-0513
- User's Handbook SX33-6061
- Messages and Codes SC33-0514

Administration

- Installation Guide SC33-0506
- Customization Guide SC33-0507
- Resource Definition (Online) SC33-0508
- Resource Definition (Macro) SC33-0509
- Operations Guide SC33-0510
- CICS-Supplied Transactions SC33-0511

Programming

- Application Programming Primer SC33-0139
- Application Programmer's Reference SC33-0512

Service

- Problem Determination Guide SC33-0516
- Diagnosis Reference LC33-0517
- Diagnosis Handbook LX33-6062
- Data Areas LC33-0518

Special Topics

- Intercommunication Guide SC33-0519
- Recovery and Restart Guide SC33-0520
- Performance Guide SC33-0521
- XRF Guide SC33-0522

Version 1 books

Application Programmer's Reference Manual (Macro Level) *SC33-0079*

IBM 3270 Data Stream Device Guide *SC33-0232*

IBM 4700/3600/3630 Guide *SC33-0233*

IBM 3767/3770/6670 Guide *SC33-0235*

IBM 3650/3680 Guide *SC33-0234*

IBM 3790/3730/8100 Guide *SC33-0236*

Books from Related Libraries

MVS

For general information on System/370, see:

MVS/XA Installation: System Generation, GC26-4009
Installation: System Generation Reference, GC26-4009
System Programming Library: System Macros and Facilities Volume 1, GC28-1150
System Programming Library: System Macros and Facilities Volume 2, GC28-1151
(or GB0F-1014 to order both volumes)
MVS Supervisor Services and Macro Instructions, GC28-0683.

For reference and information on deleting, renaming, and protecting files, see:

Data Administration Macro Instructions Reference, GC26-4014.

For reference information on space allocation, see:

OS/VS2 MVS Data Management Services Guide, GC26-3875
MVS/Extended Architecture Data Administration Guide, GC26-4013.

For access method logic, see:

OS/VS2 SAM Logic, SY26-3832
OS/VS2 BDAM Logic, SY26-3831
OS/VS2 BTAM Logic, SY27-7246
OS/VS2 MVS VTAM Logic, SY28-0621
MVS/XA SAM Logic, LY26-3891.

IMS/VS

If you use the CICS-DL/I interface, see the following IMS/VS manuals:

IMS/VS Version 1

IMS/VS Release Guide, SH20-9207
IMS/VS General Information Manual, GH20-1260
IMS/VS Installation Guide, SH20-9081
IMS/VS Primer Function Installation Guide, SH20-9208
IMS/VS Application Programming, SH20-9026
IMS/VS Application Programming for CICS/VS Users, SH20-9210
IMS/VS Application Programming Reference Summary, SX26-3727
IMS/VS Data Base Administration Guide, SH20-9025
IMS/VS System Administration Guide, SH20-9178
IMS/VS Message Format Service User's Guide, SH20-9053
IMS/VS Utilities Reference Manual, SH20-9029
IMS/VS System Programming Reference Manual, SH20-9027
IMS/VS Programming Guide for Remote SNA Systems, SH20-9054
IMS/VS Operations and Recovery Guide, SH20-9209
IMS/VS Operator's Reference Manual, SH20-9028

IMS/VS Data Base Recovery Control: Guide and Reference, SH35-0027
IMS/VS Messages and Codes Reference Manual, SH20-9030
IMS/VS Failure Analysis Structure Tables (FAST) for Dump Analysis, LY20-8050
IMS/VS Diagnosis Guide, LY20-8063
IMS/VS Diagnosis Reference, LY20-8069
IMS/VS Data Base Recovery Control: Logic, LY35-0028
IMS/VS Master Index and Glossary, SH20-9085.

IMS/VS Version 2 Releases 1 and 2

IMS/VS General Information Manual, GC26-4180
Introducing the IMS/VS Library, GC26-4294
IMS/VS Installation Guide, SC26-4172
IMS/VS Application Programming, SC26-4178
IMS/VS Application Programming for CICS/VS Users, SC26-4177
IMS/VS Application Programming Reference Summary, SX26-3746
IMS/VS Data Base Administration Guide, SC26-4179
IMS/VS System Administration Guide, SC26-4176
IMS/VS Message Format Service User's Guide, SC26-4181
IMS/VS Utilities Reference Manual, SC26-4173
IMS/VS Customization Guide, SC26-4187
IMS/VS Programming Guide for Remote SNA Systems, SC26-4186
IMS/VS Operator's Reference Manual, SC26-4175
IMS/VS Data Base Recovery Control: Guide and Reference, SC26-4209
IMS/VS Messages and Codes Reference Manual, SC26-4174
IMS/VS Failure Analysis Structure Tables FAST for Dump Analysis, LY26-3992
IMS/VS Master Index and Glossary, SC26-4182
IMS/VS Installation Listings, SC26-4215
IMS/VS System Definition Reference, SC26-4216
IMS/VS Licensed Program Specifications, GC26-4171
IMS/VS Summary of Operator Commands, SX26-3754.

IMS/VS Version 2 Release 1

IMS/VS Release Guide, SC26-4185
IMS/VS Operations and Recovery Guide, SC26-4183
IMS/VS Data Base Recovery Control (DBRC) Guide and Reference, SC26-4029
IMS/VS Diagnosis Guide, LY26-3991
IMS/VS Diagnosis Reference, LY26-3993.

IMS/VS Version 2 Release 2

IMS/VS Release Guide, GC26-4325
IMS/VS Operations Administration Guide, SC26-4323
IMS/VS Sample Operating Procedures, SC26-4324
IMS/VS Diagnosis Guide and Reference, LY27-9526
IMS/VS Program Organization, LY27-9527.

Systems Network Architecture

See the following Systems Network Architecture (SNA) manuals for further information on SNA:

Format and Protocol Reference Manual: Architecture Logic, SC30-3112
Sessions between Logical Units, GC20-1868
Introduction to Sessions between Logical Units, GC20-1869.

Contents

Customer Information Control System
CICS/MVS Version 2 Release 1
Diagnosis Reference

QUESTIONNAIRE

To help us produce books that meet your needs, please fill in this questionnaire. It would help us if you provide your name and address in case we need to clarify any of the points you raise. Please understand that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

1. Please rate the book on the points shown below
   The book is:

accurate	1	2	3	4	5	inaccurate
readable	1	2	3	4	5	unreadable
well laid out	1	2	3	4	5	badly laid out
well organized	1	2	3	4	5	badly organized
easy to understand	1	2	3	4	5	incomprehensible
adequately illustrated	1	2	3	4	5	inadequately illustrated
has enough examples	1	2	3	4	5	has too few examples
And the book as a whole?						
excellent	1	2	3	4	5	poor

2. When using this book, did you find what you were looking for? _____
   What were you looking for? _____
   What led you to this book? _____
   Did you come straight to this book? _____

3. Which topics does the book handle well?

4. And which does it handle badly?

5. How could the book be improved? _____

6. How often do you use this book?
   Less than once a month? ☐ Monthly? ☐ Weekly? ☐ Daily? ☐

7. What sort of work do you use CICS for? _____

8. How long have **you** been using CICS? ____ years/months

9. Have you any other comments to make?

Thank you for your time and effort. No postage stamp necessary if mailed in the USA. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to either address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

LC33-0517-0

Questionnaire

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1H
180 Kost Road
Mechanicsburg, PA 17055, USA

Fold and tape | Please do not staple | Fold and tape

IBM®

Name .....

Job Title ..... Company .....

Address .....

..... Zip .....

Part 1. Introduction

This part provides an overview of CICS. It introduces the functions of CICS and the more important control blocks and tables.

This part contains the following chapters:

- CICS Structure
- CICS Real-Time Execution Environment
- CICS Communication with SNA Logical Units.

Chapter 1.1. CICS Structure

CICS is logically structured into major components, as shown in Figure 1 on page 4. These components provide services to the CICS user. Most services are requested directly by the application programs through EXEC CICS commands or CICS macro instructions, but some, which help to create a more useful DB/DC environment, are performed automatically by CICS.

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

System Management Component

System management contains most of the functions that are central to the running of CICS. All the functions are resident in the CICS nucleus, or the MVS link pack area (LPA).

Task Management

Task management controls the allocation of processor time between contending CICS tasks. It is the counterpart of the central dispatcher in an operating system, providing an interface to the multitasking facilities of the host operating system. It provides some services to the application programs: attaching tasks, synchronizing tasks, and queuing for resources. It also provides certain optional control functions, for example, system stall detection, and runaway task control.

Subtask Management

The subtask management program is the interface between a CICS task and a subtask. This avoids suspending CICS execution, and improves the response time.

Storage Management

Storage management controls the virtual storage allocated to CICS 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. The services provided include multiprogramming, the loading, linking, and deletion of programs, and the transfer of control between them.

Table Management

Table management controls the locating, adding, and deleting of entries in certain CICS tables. These operations can be performed while CICS is executing.

EXEC Interface

The EXEC interface program enables an application programmer to write CICS programs using the command interface.

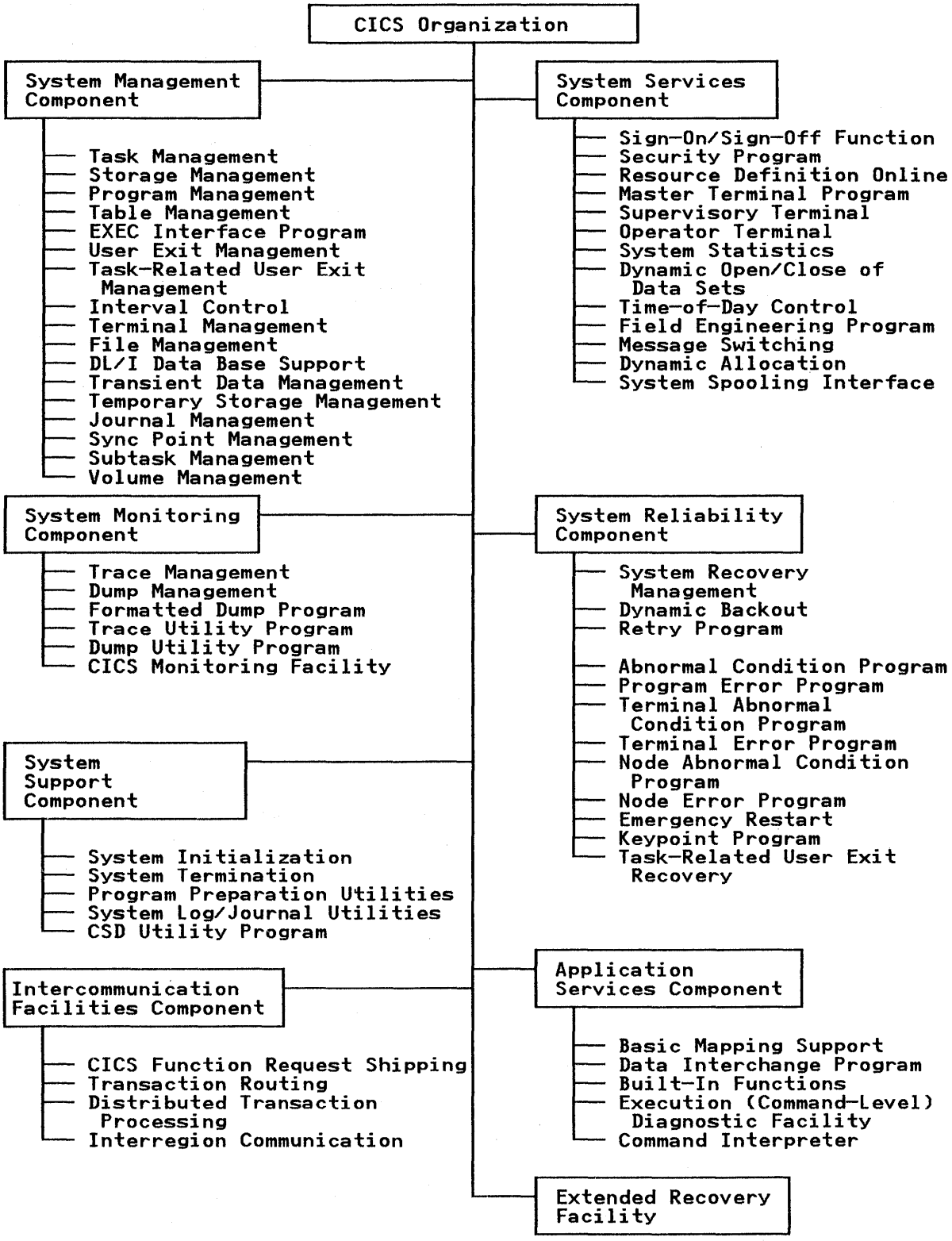


Figure 1. CICS Organization – Components and Functions

User Exit Management

User exit management enables the user to run exit programs at selected points in CICS modules without having to reassemble the relevant module. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

Task-Related User Exit Management

The task-related user exit is a means of invoking resource managers that are outside the CICS system. The application request is made through a module called a task-related user exit, so that dependence on the form of the resource manager is minimized.

Interval Control

Together with task management, interval control provides various optional task functions 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), Telecommunication Access Method (TCAM), and Virtual Telecommunication Access Method/Network Control Program (VTAM/NCP) are used for most terminal data management and line control services. Terminal management supports automatic task initiation to process transactions that use a terminal but are not directly initiated by the terminal operator. Terminal management can also provide a simulation of terminals, using sequential devices, in order to help test new applications.

File Management

File management provides a data-set (or file) access facility using keyed or relative byte address (RBA) access through the virtual storage access method (VSAM), and the basic direct access method (BDAM). File management supports updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data in the data sets. If VSAM is used, access to logical data can be made via a VSAM alternate index path, as well as via the base data set.

DL/I Data Base Support

Facilities for accessing DL/I data bases are available via the DL/I facilities of IMS/VS.

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 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 series of 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 activity in the system.

Volume Management

For standard-labeled tapes, volume management directs the choice of journal tapes to be opened.

Sync Point Management

Sync point management allows the user to specify logical units of work by means of **sync points**. Any processing performed between sync points (provided the resources are declared as recoverable) can be reversed in the event of an error; but **after** a given sync point has been reached, the processing performed **before** that sync point cannot be reversed.

A sync point is also taken automatically in the course of a transaction restart, or a DL/I TERM, and at the end of each task.

System Services Component

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 system service functions are extremely valuable to the running of an installation's DB/DC system.

Sign-On/Sign-Off Function

This function provides terminal operator identification for security purposes.

Security Program

The security program provides an optional facility for checking user authorization to run transactions and access resources.

Resource Definition Online

The CEDA transaction creates and alters the definitions of system resources in the CICS system definition (CSD) file.

Master Terminal Functions

The master terminal program provides dynamic user control of the system. Using this function, a master terminal operator can: change the status and values of parameters used by CICS and thereby alter the operation of the system; temporarily disable entries in several CICS tables; or terminate any CICS task currently in the system.

Supervisory Terminal Functions

The supervisory terminal functions provides 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 Functions

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

System Statistics

This function provides for CICS to log system statistics.

Dynamic Open/Close

The dynamic open/close function allows the user application program or master terminal to open and close data sets during the real-time execution of CICS.

Time-of-Day Control

Time-of-day control helps CICS to run continuously for more than 24 hours. CICS 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.

Field Engineering Program

The field engineering program is primarily designed to help field engineers install new terminals during the real-time execution of CICS. On request, all printable characters can be sent to the terminal, or a message can be "echoed" (that is, returned to the terminal). This program performs all operations requested by the CSFE transaction.

Message Switching

This function provides the user with a general purpose message-switching capability while CICS 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 Spooling Interface

A system programmer can communicate with the local system spooler, and, consequently, with other system spoolers via the system spooling network facilities. The system spooling interface single-threads its input, and it is the user's responsibility to see that all transactions get the chance to run. One high-priority transaction should not use the interface exclusively.

File Allocation

Any data set defined to file management can be allocated to CICS dynamically when the file is opened, rather than at CICS job initiation time. This allocation takes place automatically if job control statements for the data set are not included in the CICS job stream, and if both the data set name and the disposition have been specified in the file control table when the data set is opened.

The dynamic allocation sample program provides an alternative way to perform dynamic allocation. When used with a terminal of the IBM 3270 Information Display System, it gives the user access to the functions of DYNALLOC (SVC 99) in OS/VS2 (MVS). This can be used, in conjunction with master terminal functions and suitable operating procedures, to allocate and deallocate any file that CICS can dynamically open and close.

System Monitoring Component

The system monitoring component consists of functions that run online and provide statistics and diagnostics to the user. These functions are resident in the CICS nucleus.

Trace Management

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

Trace entries are recorded in an area of main storage called the **trace table**. In addition, trace entries can be recorded in auxiliary storage on a sequential data set (tape or disk), for subsequent offline formatting and printing, using the CICS trace utility program. This method of recording is known as **auxiliary trace**.

Dump Management

Dump management provides help in analyzing programs and transactions that are being developed or modified. Specified areas of dynamic storage are dumped onto a sequential data set (tape or disk), for subsequent offline formatting and printing, using the CICS dump utility program.

Formatted Dump Program

The formatted dump program is automatically run when a program check or system error occurs (depending on system initialization table (SIT) and program control table (PCT) parameters). It produces a dump of the CICS address space with the various CICS control tables and areas identified.

Trace Utility Program

The offline trace utility program formats and prints the contents of the auxiliary trace data set. This program operates in batch mode, and formats each trace entry to show the CICS component that made the entry and the function being performed.

Dump Utility Program

The offline dump utility program formats and prints the output from formatted dump, and prints transaction dumps. It operates in batch mode and, for formatted dumps, identifies each storage area, program and table entry, and prints them separately, with actual and relative addresses.

CICS Monitoring Facility

The CICS monitoring facility gives a comprehensive set of operational data for CICS, using one data recording program and, optionally, one or more data sets.

System Reliability Component

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 intercepts program interrupts and abnormal terminations by the host operating system and prevents them from terminating the whole CICS 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 Backout

Dynamic backout allows the effects of an abnormally terminating transaction to be reversed immediately, while the rest of CICS continues n ɔrmally.

Retry Program

The retry program is an optional facility that can be specified for individual transactions. In the event of program isolation deadlock (that is, when two tasks each wait for the other to release a particular DL/I data base segment), one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

Abnormal Condition Program

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

Program Error Program

Each CICS installation may supply a routine to provide a user action in response to a programming error. CICS 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 Programs

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

Terminal/Node Error Programs

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 distribution volume.

Emergency Restart

The emergency restart function allows users the option of restarting CICS following an abnormal termination (machine check, power failure, or abnormal termination by the operating system), and reinitializing CICS selectively in order to meet their requirements.

Recovery Utility Program

When emergency restart is executed, system initialization calls the recovery utility program to process system recovery data. The recovery utility program scans the journal data set backward and writes all data for in-flight tasks to the restart data set.

Transaction Backout Program

The transaction backout program is invoked during emergency restart, and reads data from the restart data set which has already been written by the recovery utility program. This data is backout information for task, message, DL/I and file tables.

Keypoint Programs

The warm keypoint program, as part of the restart program, performs a warm keypoint at normal termination of CICS.

The activity keypoint program writes keypoints to the system log for the current state of the system.

Task-Related User Exit Recovery

Task-related user exit recovery ensures that changes to recoverable resources performed by an external resource manager in a logical unit of work are either all committed or all backed out.

System Support Component

The system support component consists of several functions (mostly offline functions) that are required to support the CICS system.

System Initialization

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

System Termination

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

Program Preparation Utilities

Before they can be compiled or assembled, all CICS application programs (except those in assembler language using CICS macros) have to be processed by an offline utility program – either a **high-level language preprocessor** or a **command-language translator**.

High-Level Language Preprocessor

The high-level language preprocessor performs part of the process of preparing a COBOL or PL/I program that includes CICS macro instructions.

Command Language Translator

A command language translator program is used to prepare an application program that includes EXEC CICS commands. The translator program translates the EXEC commands into CALL statements in the language of the application program. The output can then be compiled (or assembled) in the usual way.

System Log/Journal Utilities

Format Tape

The format tape utility preformats magnetic tapes to be used for journals, so that the tape end-of-file utility can operate on the tape correctly if subsequently needed.

Format Disk

The format disk utility preformats a disk data set to be used for a system log or journal so that the journal control open program (DFHJCO) can locate the last record written.

Tape End of File

The tape end-of-file utility verifies tape volumes and control records, and writes an end-of-file mark on tape for emergency restart. The end-of-file mark is written when the access method finds an unreadable record, or a record that does not belong to the same part of the journal.

CSD Utility Program

The CSD utility program (DFHCSDUP) is an offline program which provides a means of creating and maintaining the CICS system definition (CSD) file. The functions performed are initialize, migrate, copy, append, erase, list, verify, and service.

Application Services Component

CICS provides several functions designed to perform services closely associated with user applications. These services rely on CICS 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 (BMS) 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 multipage 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 automatically inserts the control characters and eliminates trailing blanks from each line. Most of the BMS programs are resident in the CICS nucleus.

Data Interchange Program

The data interchange program supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.

Built-In Functions

CICS provides the application programmer with some commonly used functions, invoked through the CICS macro-level interface. The built-in function program is resident in the nucleus, and includes table search, phonetic conversion, field verify, field edit, bit manipulation, input formatting, and weighted retrieval.

Execution (Command-Level) Diagnostic Facility (EDF)

The execution diagnostic facility (EDF) helps users of the CICS command-level programming interface by allowing them to step through the CICS commands of an application program. At each step, the user can check the validity of each command and make temporary modifications to the program.

Command Interpreter

The command interpreter assists the user in writing syntactically correct CICS commands and shows the results of execution of a command entered on a display.

Temporary Storage Browse Transaction

The temporary storage browse transaction allows the user to browse, copy, or delete items in a queue.

Intercommunication Facilities Component

CICS Function Request Shipping

The CICS function request shipping facility enables an application program to make a request for a resource in a different CICS system and have that request shipped to the system that controls the resource, without knowing where the resource is. The other CICS system can be in a different address space of the same processor system or in a different processor system.

Transaction Routing

Transaction routing is the ability of a terminal in one CICS system to initiate a transaction in another CICS system. The application program need not be aware that the terminal it is accessing is owned by a different CICS system.

Distributed Transaction Processing

Distributed transaction processing enables a CICS application program to initiate transaction processing in a system which resides in the same or a different processor system or in different address spaces of the same processor. Distributed transaction processing is invoked by terminal control commands which acquire a session and allow synchronous conversations between application programs.

Interregion Communication

Interregion communication is the mechanism used to communicate between CICS systems (MRO), or DL/I batch regions and CICS systems (shared data base) running under a single MVS system. For shared data base, interregion communication allows an IMS/VS batch job to access a data base owned by a CICS system in the same processor. The CICS DL/I batch region controller handles requests by an IMS/VS batch application program instead of the IMS/VS data base batch region controller.

For shared data base and multiregion operation (MRO), the MVS cross-memory services routine or the SVC routine handle the transfers of data between regions.

Extended Recovery Facility

The extended recovery facility (XRF) enables you to achieve a high level of availability. You can run an alternate CICS system which monitors your active CICS system, and takes over automatically, or by operator control, when the active system fails. You can also plan and execute a takeover yourself when you want to do maintenance on an active system.

Problems in the active system can be detected and isolated as soon as they occur. The alternate system can recover and restart quickly, like an emergency restart, and the time for reconnection of terminals is reduced.

Chapter 1.2. CICS Real-Time Execution Environment

The purpose of this chapter is to describe the way CICS supports an application program. Figure 2 shows, in a simple way, how CICS executes between an application program and the operating system. Figure 3 on page 14 shows the CICS system in more detail, and the diagram could be extended by adding all the other CICS functions that are invoked through the application interface.

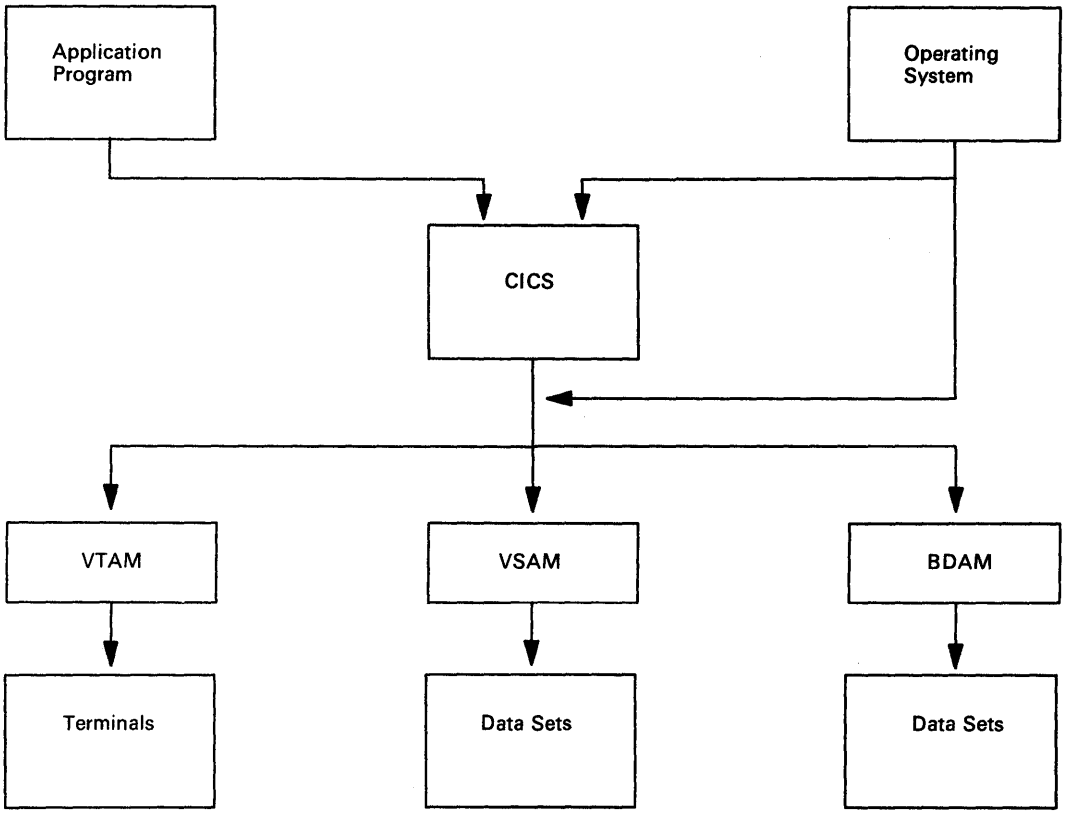


Figure 2. CICS in Context

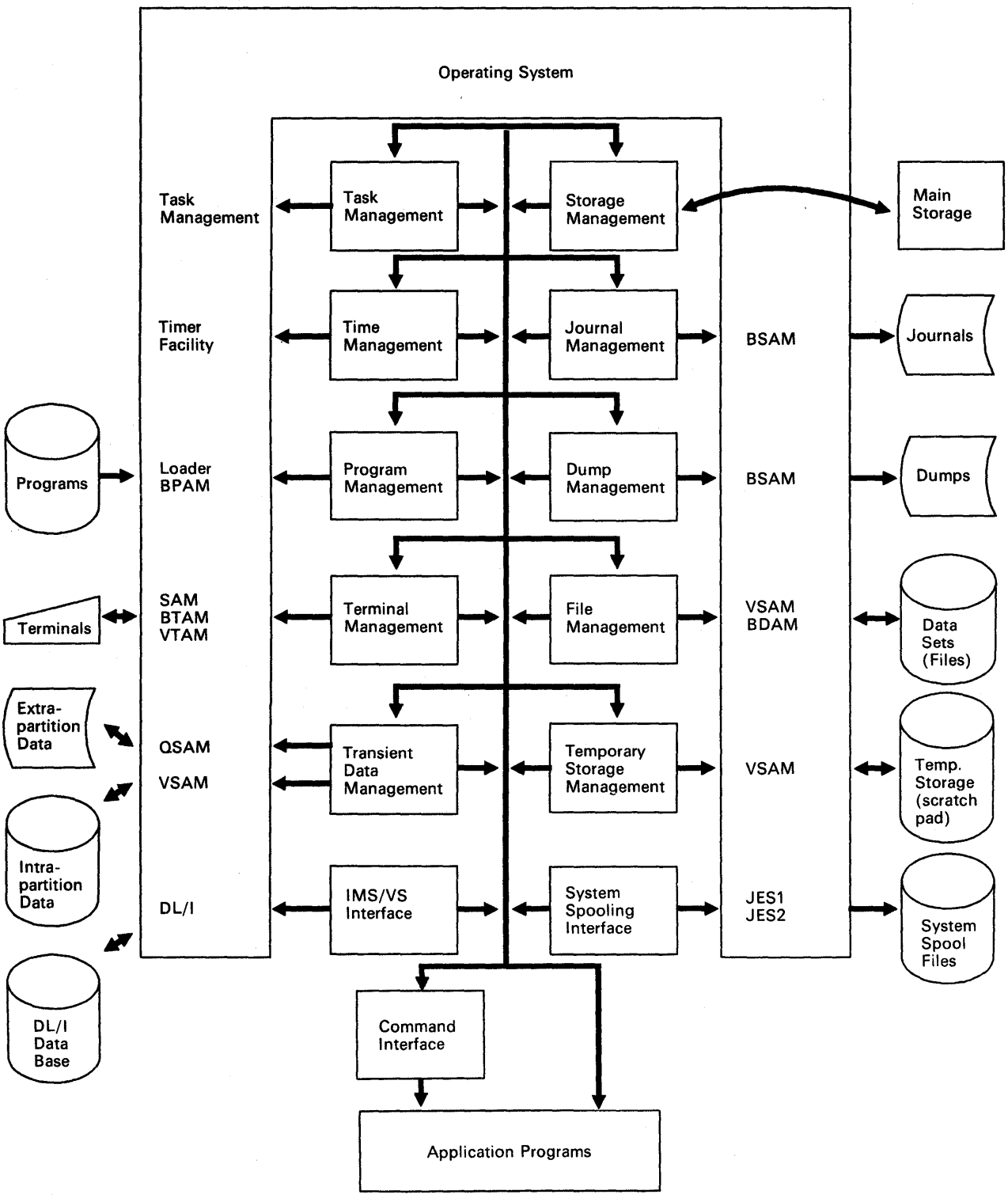


Figure 3. CICS and the Operating System

Application Interface

Application programs are written in assembler language, COBOL, or PL/I. There are two interfaces with CICS:

- The command-level interface. This is available to the assembler, COBOL, or PL/I programmer. A command-language translator converts the CICS commands to statements in the language of the program.
- 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 high-level language preprocessor that makes the macro-level interface available to the COBOL or PL/I programmer.

Command-Level Interface

The command language translator expands CICS commands into calls (from the code generated by the compilers) to an interface program. This sets up control blocks according to the command parameters, and hands control to CICS. The command-level interface is complete, so that the application programmer does not need to set fields in control blocks.

The only control block that the command-level programmer need be concerned with (apart from the user interface block (UIB) when using DL/I) is the EXEC interface block (EIB).

Macro-Level Interface

The macro-level interface can 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 macros.

For a typical macro expansion, fields may be set up in a control block according to parameters of the macro, and a call made to an address that is held in a field of the CSA. After the call, the value of the return code is tested.

The expansions of CICS macros assume that adequate addressability 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 entry point. There is usually one entry point corresponding to each macro. The entry point addresses are held in the CSA.

Further analysis of the macro invocation takes place within the CICS routine. For instance, in the example expansion above, 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 cannot be given entirely by the macros; some of it must be passed through fields in control blocks, which are examined by the CICS 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.

Intermodule Communication

CICS modules communicate with one another, where possible, by using the CICS application program interface. This is appropriate for calls to those modules, such as task management, that provide services to application programs, and many CICS macros have extensions so that they can be used to call for services that are internal to CICS. 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 modules. Parameters are implicit, because they are fields within the CICS tables and work areas.

Control Blocks

The EXEC command-level programmer does not use any of the control blocks listed here. These control blocks are used only by the macro-level programmer.

Common System Area (CSA)

The common system area (CSA) is the major CICS control block. It is an area in main storage that exists within the system from initialization time until the CICS is closed down. The CSA contains:

- Register save area
- Pointers to the CICS management modules
- Control information
- System constants
- Time-management storage
- Work area for statistics
- Task abnormal termination interface
- Pointers to CICS 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).

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

Dispatch Control Area (DCA)

A dispatch control area (DCA) is created for each task by CICS at task initiation. 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 processing or disk I/O. These chains are addressed through the CSA. The DCA is logically part of the TCA, but is kept separately to reduce page faults.

Task Control Area (TCA)

When a task is first dispatched, CICS obtains storage for the following:

- System area
- CICS application program communication section (user area)
- Transaction work area (TWA), which is optional
- Monitoring area, which is optional
- Load list area
- EIS storage (optional)
- Table management lock block storage
- LIFO (last-in/first-out) storage area.

TCA – System Area

The CICS system area of a TCA contains the addresses and data necessary for CICS to control a task. Access to data in this area is limited to CICS management programs, CICS service programs, and user-written service programs.

TCA – Application Program Communication Section (or User Area)

The application program communication section of a TCA is used for communication between the task and CICS management programs and service programs. Access to this section is provided to both CICS and user-written application programs. It is addressed through register 12. Certain fields in this area form part of the macro-level application program interface. They are listed in the *CICS/VS Application Programmer's Reference Manual (Macro Level)*.

TCA – Transaction Work Area (TWA)

The TWASIZE keyword of the CEDA DEFINE TRANSACTION command determines whether a TWA is appended to the TCA. The size of the TWA (if there is one) is specified in this keyword. The TWA is acquired 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 to pass values between programs at different logical levels. The TWA is reserved for the exclusive use of the application program.

Monitoring Area

The CICS monitoring facility uses this area in the TCA to record task-related information. The information is accumulated during the lifetime of a task, and consists of clocks, counts, and character strings that are defined by the CICS monitoring facility and monitoring control table (MCT). The size of this area is determined at task attach time. This means that any classes of monitoring that are enabled during the lifetime of a task will not affect the classes of monitoring active for that task.

Load List Area

The program control program uses the load list area to keep track of application programs loaded by the task.

EXEC Interface Storage

The EXEC interface storage is used to pass information between command-level applications and CICS.

Lock Block Storage

The table management program (DFHTMP) uses the lock block storage to keep track of which table entries a task has referenced.

Last-In/First-Out (LIFO) Storage Area

Last-in/first-out (LIFO) storage space (analogous to automatic storage in PL/I) is used as work space and to save registers for those CICS modules invoked by the task that are classed as reentrant. When such a module is invoked, a portion of the LIFO storage space (called a LIFO stack entry) gets allocated as a data area for use by that particular module. If that module in turn links to another reentrant CICS module, a further stack entry is allocated.

Stacking continues in this way for as long as necessary, that is, until a linked-to module completes execution. At this time, the LIFO stack entry for that module is released. (Being the last in, it becomes the first out.) The stack entries for the other, uncompleted, modules are still intact; a stack entry is released only when the owning module runs to completion.

Use of the TCA

For tasks attached by CICS, a TCA is created when a task is first dispatched, but the TCA is created at attach time for tasks attached by macro-level applications. The pointer to the TCA always addresses the application program communication section of the TCA.

Information about terminal management 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 dispatched by task management. A separate TCA is set up for this task, so that it can request services of CICS 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 nonterminal storage can be returned to storage management as soon as the task ends.

Automatic Initiate Descriptor (AID)

Automatic task initiation is possible under CICS. An automatically initiated task may be associated with a terminal. CICS employs a queuing technique in order to initiate a task when its terminal becomes available. An automatic initiate descriptor (AID) is created for each request for automatic task initiation and is added to the chain of AIDs. This chain is sequenced by symbolic transaction identification within symbolic terminal identification. The TCT contains the address of the first entry of the AID chain at TCSESUSF.

When an AID is added to the chain (other than for a restarting transaction), 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 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 interval control. These ICEs are logically chained from the CSA in expiration time-of-day sequence.

Expiration of a time-ordered request is detected by the expired request logic of interval control, running as a CICS 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 next gains control.

Interval control passes expired ICEs that represent either time-ordered task initiation requests, or time-ordered data records, to task management which treats them 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 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.

When BMS messages have not been delivered in a specified time, the undelivered messages cleanup program (DFHTPQ) purges ICEs that were created to deliver the messages.

Journal Control Area (JCA)

The journal control area (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 file work area (FWA) is the area in which file records are normally passed between CICS and the application program. It is acquired dynamically from main storage by file management: when updating; when reading for inquiry a blocked record; when adding a new record to a file; or when retrieving records using the browse facility.

File Input/Output Area (FIOA)

The file input/output area (FIOA) is acquired dynamically from main storage by file management whenever a request is made for access to a BDAM data set. The data area is used as the true I/O area, to and from which records are written and read. For all BDAM operations, except read without update (inquiry of unblocked records), records are moved between the FIOA and the FWA.

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.

DWEs are also used to save information for backing out if an abend or sync point rollback request should require it.

Temporary Storage Input/Output Area (TSIOA)

The temporary storage input/output area (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 TCTTE contains the address of the first terminal-type storage area obtained for a task (the beginning of the chain), and 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 header (or its copy in the trailer) is referred to as the storage accounting area (SAA).

Dynamic Buffer Area (DBLDS)

A dynamic buffer area (DBLDS) is acquired by the journal control program for use as a dynamic log for transactions for which dynamic backout is applicable. The records (DBRDS) written to this log are equivalent to those written to the system log for use by the transaction backout program during emergency restart. If the dynamic buffer area is too small, it overflows into virtual storage or temporary storage and is reused.

Command List Table (CLT)

A command list table (CLT) is used by an alternate system when it takes over the running of CICS from an active system. It holds the ID data for the JES system in use, data used to verify its authority to take over, and routing information. If there is more than one active system in two CECs, the CLT also holds VTAM MODIFY commands, and messages to the operator (WTO) to complete the takeover. It is loaded during takeover, and deleted when processed.

CICS Execution

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

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

- **Data Transfers**. Some of the active peripheral devices are copying data from the CICS address space, others are recording data in the CICS address space.

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

Thus, the system is highly dynamic and the concurrent activities described above compete with each other for the following system resources: processor time and main storage; channel, control unit, and storage device services; and communication lines.

Transactions and Tasks

We make the following distinction between the terms **transaction** and **task**.

A **transaction** is a set of CICS application programs consisting of an initial program named in a program control table (PCT) entry and any other programs called by that program.

A **task** is the execution of a transaction.

CICS Recovery

CICS protects resources and transactions in the following ways:

- 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 command that makes, or leads to, a modification of the resource (for example, READ, WRITEQ TD, READQ TS).
- 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 READ UPDATE gets and logs the **before-image** of a record to be modified. When the record is subsequently about to be updated by a WRITE, the logged **before-image** may still be resident in a journal control buffer. This buffer is written to the log before the WRITE is executed. A copy is also retained in a main storage dynamic buffer for use by dynamic backout.
- Certain irreversible operations (for example, DELETE, DELETEQ TD, or DELETEQ TS, or transmission of a protected message) are deferred until the logical unit of work (at a sync point or at end of task) is complete.

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

1. A record is made in a deferred work element (DWE) of the function to be performed and the data needed.
2. A DWE processor is activated to execute the operation after the task has issued its final RETURN, or SYNCPOINT.

Each CICS 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 immediately before the task is detached.

Figure 4 on page 22 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 on page 22) in a protected data set.

The vertical bracket alongside 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. Because journal management, which does the logging but is not shown in the diagram, is a CICS subtask, impact of the extra wait on performance is minimal. There may be no task wait time at all if the journaling 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.

If the resource is handled by a program that is external to CICS, then recovery is performed via the task-related user exit interface.

Long-Running Tasks, Sync Points, and Logical Units of Work

The sync point illustrated in Figure 4 on page 22 is taken as a result of the execution of a sync point macro instruction by task management at task detach time. A sync point can also be taken by an application program executing a SYNCPOINT command.

A user sync point forces the logging, deferred processing, and dequeueing activity that is normally not done until after a task completes (refer to Figure 4 on page 22).

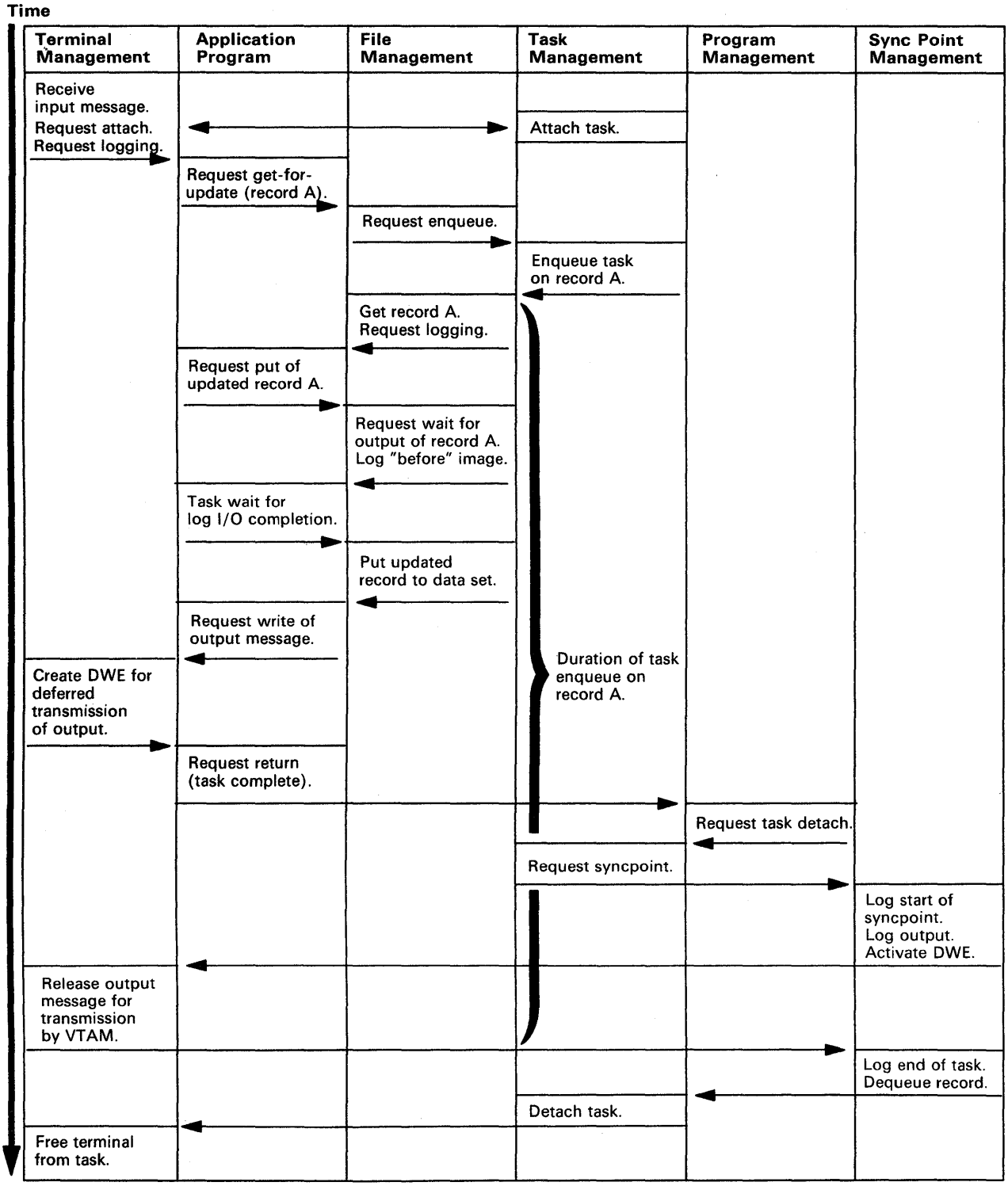


Figure 4. Flow of Control for a Simple Transaction (Including DWE Processing)

To minimize resource conflict and delays due to deferred processing, the application programmer can divide long-running tasks into logical units of work (LUWs), by means of user sync point requests, as frequently as the logic of the application allows. User sync points reduce the amount of work that has to be backed out (and redone) during emergency restart or dynamic backout (and transaction restart).

None of the work done by a task before a sync point is taken can be backed out during CICS emergency restart (or dynamic backout). This should be remembered when specifying a sync point to delimit an LUW.

Multiprogramming, Multitasking, and Multithreading

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

Multiprogramming

The CICS system as a whole may share the computer and operating system with batch programs (or even with other CICS systems or other DB/DC systems) in other address spaces. If CICS is executed in such a multiprogramming environment, it is usually in the highest priority address space (except for operating system components). Batch address spaces receive control from the operating system only when CICS has no dispatchable transactions. Thus, as long as there is a transaction ready for processing, CICS maintains system control. Control is released to allow the operating system to continue a job in another address space only when there are no more transactions ready. CICS regains control as soon as any previously waiting CICS 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 executes in a multitasking mode of operation within one or more address spaces of main storage. Multitasking within one address space 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 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 address space is initiated or continued.

When only one terminal is active, control returns to the application program. 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 can overlap the processing of one task with the I/O operations of others.

Certain CICS modules can use the subtask management program, DFHSKP, to handle their event management. This increases the overlap when operating system waits occur, and improves the response time in a CICS system.

Multithreading

In CICS, 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 the point reached by each task within a program, or the point at which the task should resume processing when it again receives control.

Storage

CICS Address Space

A map of virtual storage for a CICS system with MVS/XA is shown in Figure 5. For a description of each area, see the *CICS/MVS Performance Guide*.

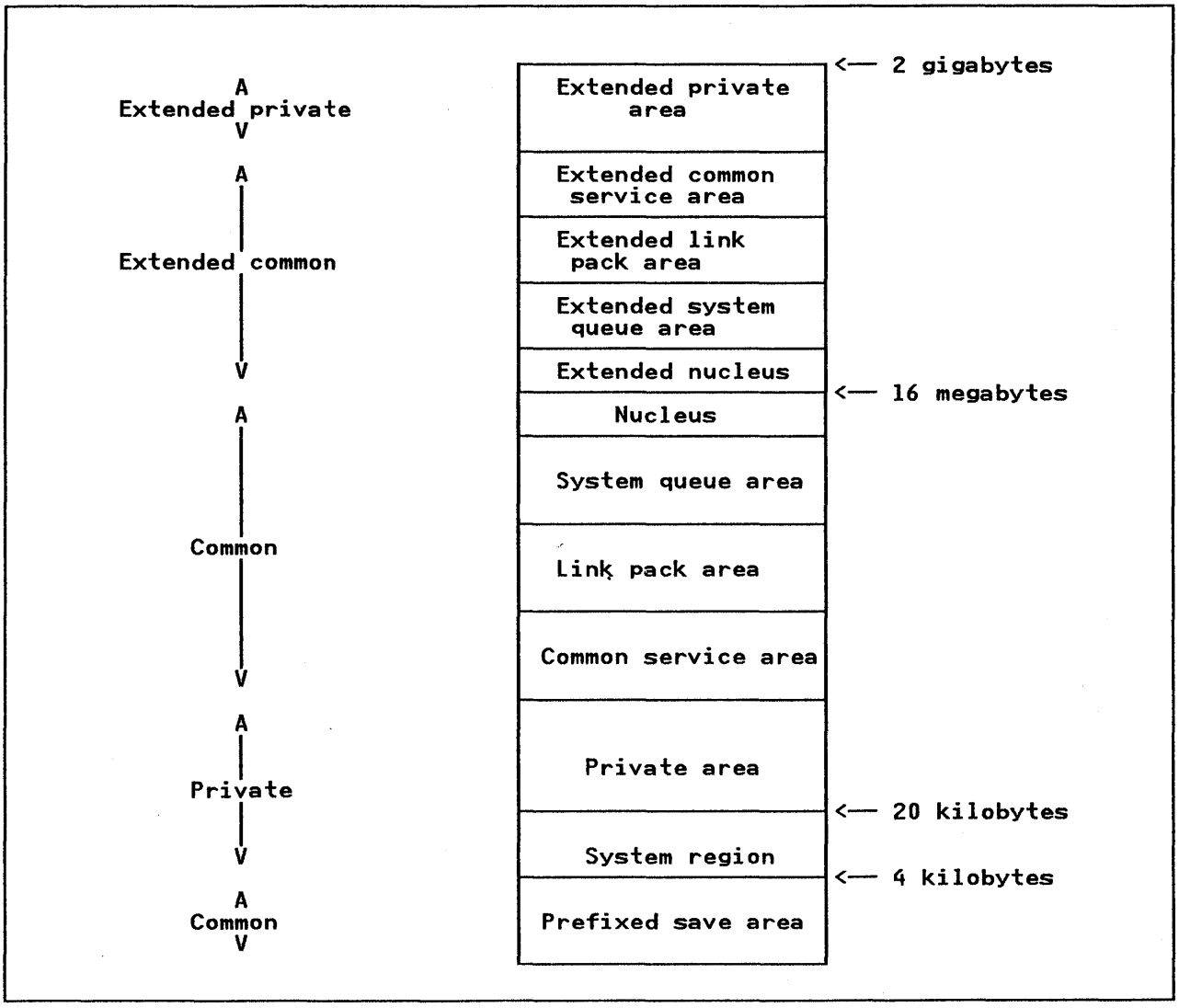


Figure 5. Virtual Storage Map for MVS/XA

Allocation of Dynamic Storage by Subpools

CICS dynamic storage space is controlled by storage management, which allocates storage, depending on the type of request, from different **subpools**. (A subpool is a set of virtual storage page frames.) Each subpool obtains and returns storage in page multiples from the dynamic storage area, which is described in the CICS page allocation map (PAM). The address of the PAM is kept in the CSA.

The types of subpools are:

- Task subpool
- Program subpool
- Control subpool
- Teleprocessing subpool
- RPL subpool
- Shared subpool.

The task subpool is used for all transaction-related storage. CICS allocates complete pages and frees them, as required, on a task basis. There is no mixing of storage areas of different tasks on the same page.

All dynamically loaded application programs are allocated contiguous full page frames in the **program subpool**. Any program less than one full page (2K bytes or 4K bytes, depending on the SIT PGSIZE operand) 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 small areas of storage needed by CICS 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 that are waiting on the availability of a particular resource. A **teleprocessing subpool** is used for line and terminal input/output areas.

The **RPL subpool** is used for the storage needed by CICS in VTAM terminal management.

The **shared subpool** is used for CICS storage areas that are not associated with a particular task, but are too large to use the control subpool.

In order to speed up storage requests, 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 reuse by a new transaction of the same type.

Data Sets

System Data Sets

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

- CICS program library
- CICS system definition (CSD) data set
- Restart data set
- Dump data set
- Intrapartition transient data set
- Temporary storage data set
- System log data set
- Automatic statistics data set
- Auxiliary trace data set.
- CAVM control data set
- CAVM message data set

System data sets, other than the dump data set, system log data set, and auxiliary trace data set, must be located on direct-access storage. The dump data set, system log data set, and auxiliary trace data set can 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 options selected at CICS

system generation and the extent to which they are used.

CICS Program Library

The CICS program library contains all user-written programs and CICS programs to be loaded and executed as part of the online system. This group includes the control system and certain user-defined system control tables essential to CICS 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 dynamic storage for online execution.

CICS System Definition File (CSD)

The CICS system definition file (CSD) is a VSAM KSDS cluster. The CSD data set holds definitions of entries in the PCT, PPT, and VTAM and MRO terminals and sessions in the TCT.

Restart Data Set

The restart data set (a VSAM KSDS cluster) is used by the CICS warm keypoint program (DFHWKP) to save certain system environment information at system termination so that CICS can be warm started later if necessary.

The restart data set consists of two logical files:

- The recovery file which contains the relevant recovery information collected from the system log during emergency restart.
- The catalog file which contains information for each resource manager.

This optional facility can be invoked to warm start the following CICS 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 addresses for temporary storage auxiliary destinations and queues.
- Temporary storage space allocation bit map.
- Interval control elements (ICEs) and automatic initiate descriptors (AIDs).
- Control records of identifiers of standard-labeled journal tapes in the system log. The restart data also maintains the series definition table across all starts.
- Indicators which determine whether an AUTO start is executed as a WARM start or an EMERGENCY restart.

You also need a restart data set if you are using journaling.

Dump Data Set

The dump data set (optional) is used by dump management to record dumps of tasks within the system. It is organized sequentially, and can be formatted and printed by the CICS dump utility program. If required, the user can define two dump data sets (DFHDMPA and DFHDMPB), alternating between them during online execution of CICS. The data set can be either tape or disk.

Intrapartition Transient Data Set

The intrapartition transient data set (optional) can be used for queuing user data and CICS 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 (optional) is required only when the general-purpose scratch pad or queuing facilities of temporary storage management are utilized, if the time-ordered automatic task initiation facility of CICS 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 is used for logging activity on protected resources. During an emergency restart, the recovery utility program reads the system log backward 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 control program (DFHISTSP) to record system statistics. The data set (DFHSTN or DFHSTM) is organized sequentially, and is located on either tape or disk. 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 organized sequentially. The trace utility program (DFHTUP) can be used to print records from this data set.

CAVM Control Data Set

The CAVM control data set holds a control interval (CI) for recording the state of the generic (GEN), and a status CI for the systems in an XRF complex.

CAVM Message Data Set

The CAVM message data set holds secondary status control intervals (CIs) to be used when a system is unable to write its status CI in the control data set, and enough message CIs to hold the maximum anticipated backlog of messages from the active system to the alternate system in an XRF complex.

User Data Sets

User data sets comprise those data sets that form the CICS 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, DL/I data sets and journal data sets. The user data sets are:

- File management user data sets
- Extrapartition transient data sets
- Terminal management sequential data sets
- DL/I data bases
- Journal data sets
- CICS monitoring facility data sets.

User Data Sets Managed by File Management

User data sets for file management must be placed on direct-access storage and can be accessed under control of VSAM, or BDAM. Under either of these access methods, the user has a great deal of flexibility in defining the structure of the data sets.

Extrapartition Transient Data Sets

Extrapartition transient data is usually routed to or from high-speed input/output devices and typically consists of blocked, variable-length records. Extrapartition transient data sets are used with data collection and data entry applications, and for output of data that will be processed later (usually offline).

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.

DL/I Data Bases

CICS accesses DL/I data bases using of the 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).

Journal Data Sets

If CICS journal management is used, entries in the CICS journal control table (JCT) identify user journals and/or the system log. These consist of separate sequences of data sets which can be on either tape or disk. Each journal must be given a journal identification.

CICS Monitoring Facility Data Sets

CICS monitoring facility data sets are used to record information that is generated by the CICS monitoring facility program. The MCT defines which journal data sets are used by each class of monitoring. These data sets appear in the JCT with the keyword FORMAT = SMF. The format of the records is the system management facility (SMF), type 110 format.

Chapter 1.3. CICS Communication with SNA Logical Units

This chapter provides an overview of the operation of CICS when it is being used to communicate in a systems network architecture (SNA) network with SNA logical units.

A logical unit in an SNA system may be:

- A terminal.
- One of several functions of a terminal subsystem (for example, the IBM 3790 Communication System).
- Another host system (for example, CICS).

CICS communicating with an SNA communication subsystem requires the following components:

- CICS and its application programs in the host processor.
- A telecommunications access method (VTAM, or TCAM).
- An operating system (MVS/XA).
- Unless the subsystem is locally connected, one or more communications controllers and Network Control Program/Virtual Storage (NCP/VS).
- One or more SNA communication subsystems, including at least one terminal for the control operator, and possibly a subsystem application program.

When requesting communication with a subsystem in an SNA communication subsystem, CICS application programs use the same types of request (at command-level or macro-level) that are used to communicate with any other terminal. Requests involving an SNA communication subsystem may be sent through a **communications controller** and the CICS application program communicates with **logical units** (which include application programs, control programs, or terminals).

CICS application programs can use terminal control, BMS, or data interchange commands or macro instructions to transfer data to or from the logical units. Data is transferred between the CICS application programs and CICS, by means of a terminal input/output 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 by DFHTCT macro instructions, or the resource definition online (RDO) facility.

Some subsystems are programmable, and the logical units may correspond to subsystem application programs. These subsystem application programs communicate with CICS application programs, and, in this case, the characteristics of the logical unit in the subsystem, as it appears to CICS, will depend on the way the corresponding subsystem application program is written.

The basic operation of CICS communicating with an SNA communication subsystem involves:

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

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

1. The operating system of the host processing unit is loaded.

2. CICS and VTAM are started. Communication between CICS and VTAM is established by CICS issuing an OPEN VTAM ACB request.

   - If VTAM is started before CICS, then the initialization of CICS will open the VTAM ACB automatically.

   - If VTAM is started after CICS, the VTAM ACB can be opened only by a CICS master terminal (CEMT or CSMT) command.

   Subsequently, the VTAM ACB can be closed and opened dynamically by CEMT or CSMT commands.

3. Depending on options specified, starting VTAM can also cause the activation of NCP/VS and/or subsystems. These items can alternatively be activated individually by the VTAM network operator.

4. The Synchronous Data Link Control (SDLC) cluster controller (if present) is initialized for the SNA communication subsystem. This initialization causes the designated operating environment to be loaded into the SDLC cluster controller.

5. Assuming that the VTAM ACB is open (as described in step 2 above), CICS can then connect CICS application programs with logical units and enable them to communicate with each other.

Figure 6 shows the interconnections between the components of a communication system.

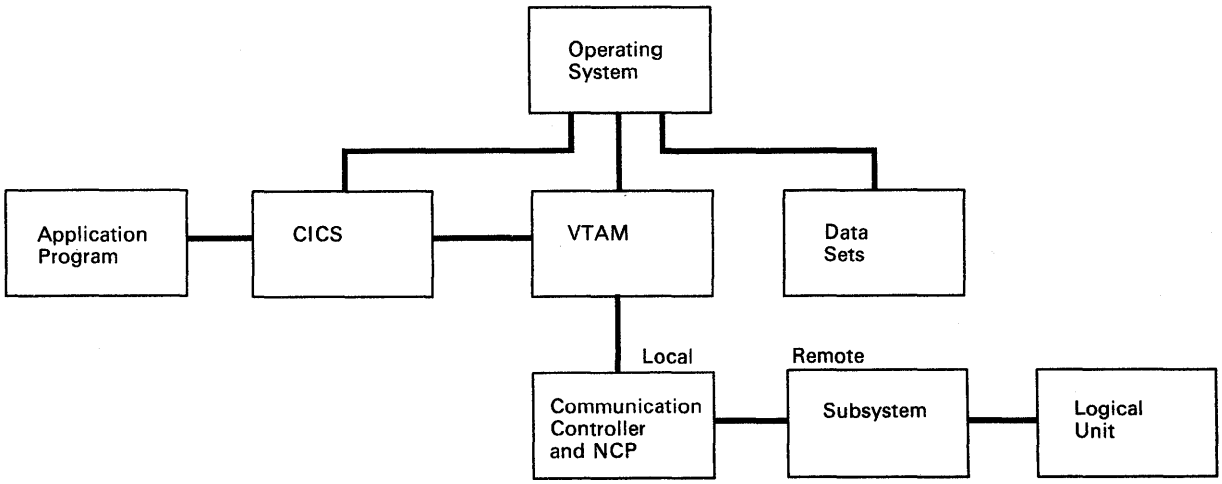


Figure 6. Interconnections between CICS System and an SNA Communications Subsystem

Sessions

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

A logon is the only method by which a session between a logical unit and CICS can be initiated. Logon may be initiated by the logical unit, by the network operator, by the CICS master-terminal operator, automatically by the access method, or by CICS itself. CICS 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.

Logical units for which CICS is to initiate logons are identified to CICS through the terminal control table (TCT). CICS also uses the SIMLOGON macro instruction to obtain logical units that are

requested by the master-terminal operator or that are invoked in an automatic task initiation (ATI) but are not currently connected.

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

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

CICS 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. (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.

On receipt of the connection request and the first message from a logical unit, CICS 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 send data must also include a begin-bracket designation. Bracket protocol is used for transactions between CICS application programs and most types of logical unit; this protocol delimits the CICS transaction.

Once in session with CICS, the logical unit can initiate any number of CICS 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. There are two types of session termination: orderly and immediate.

Orderly Termination of a Session

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 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.

CICS initiates orderly termination by issuing the shutdown indicator. The subsystem recognizes the shutdown indicator and returns a signal to CICS. CICS 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 of a Session

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

A request for immediate termination can be initiated by:

- The logical unit closing the session
- CICS 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 a terminate indicator to CICS.

CICS 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 on behalf of the logical unit after receiving the clear-unbind sequence.

Sign-Off

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

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

If you use CSSF LOGOFF, the session terminates only if DISCREQ = YES is specified in the TYPETERM definition.

Data Transmission

While a transaction is in progress, the CICS application program uses CICS commands to request CICS services (including data transfer services for the logical unit). CICS, 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 SNA communication subsystem application program for the designated logical unit uses subsystem instructions to request the services of the SNA 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 command to use 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 application program and the logical unit. All data exchanged is in a user-defined format.

Reading Data from a Logical Unit

To read data from a logical unit, the application program issues a RECEIVE or CONVERSE command.

When a task is attached by CICS, the input data is presented to the CICS 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 application program in a terminal I/O area (TIOA) addressed through the TCTTE.

When a CICS application program issues a RECEIVE command 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.

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 VTAM-supported start-stop or BSC terminal, CICS may ignore the data, which will then be lost.

If, however, unexpected data arrives from an IBM 3600 Finance Communication System, an IBM 3650 Retail Store System, an IBM 3767 Communication Terminal, or an IBM 3770 Data Communication System interactive (contention only) logical unit, or an IBM 3790 Communication System inquiry logical unit, it is queued, and used to satisfy any future input requests for that logical unit. For the 3270 logical unit (but not for the 3270 LUTYPE2 logical unit), data is queued only if PUNSOL = NO is specified in the DFHSG PROGRAM = TCP macro; otherwise, it is lost. Unsolicited input does not occur for the other logical units.

Inbound Function Management Header (FMH)

The CICS 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 can 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. The programmer 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 the programmer specified that all inbound FMHs will be passed to the application program, the INBFMH operand of the DFHPCT TYPE=ENTRY macro instruction should be coded. This operand instructs CICS 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 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 application program may be communicating with a logical unit instead of with a terminal, it still uses the SEND command to transmit output data to its destination. At the time of issuing the SEND command, 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 CONVERSE command expands into SEND, WAIT followed by a RECEIVE, 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.

Control is returned to the application program following completion of the SEND, RECEIVE sequence. The address of the input data is found in TCTTEDA; the length of the input data is specified by TIOATDL.

Synchronization of Terminal I/O (WAIT)

It is not essential to use the WAIT option or command. CICS infers a WAIT before a subsequent SEND, RECEIVE, SYNCPOINT or FREE command.

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 SEND command exceeds the RU size, CICS automatically breaks up the data into RUs and transmits these RUs as a chain. During transmission from CICS 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 SEND request. The data supplied in the TIOA for the next SEND request will then be treated as a continuation of the chain. To accomplish this chain-building function, the SEND request must include the CCOMPL=NO 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 or by the CICS application program. If built by CICS (for the 3600 Finance Communication System only), the CICS application program must be sure to reserve the first three bytes of the message for the FMH.

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

Bracket Protocol

Bracket protocol can be used when CICS communicates with a subsystem logical unit. For the most part, the use of brackets is not apparent to the CICS application program.

Only on the last SEND operation to a logical unit does the bracket protocol become apparent to the application program. On the last output request to a logical unit, the application program may specify LAST in the SEND command. The LAST specification causes CICS 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 waits until the task detaches before sending the end-bracket indicator.

Data Chaining

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

CICS 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 may have already sent the rest of the chain; therefore, the subsystem must purge the rest of the chain. When a cancel indicator is received, the subsystem should ignore the data chain currently being received and exit from 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 analogous 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 to schedule DFHZNAC. DFHZNAC is scheduled for a TCTTE any time that an operation requested by CICS 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, the node error program (DFHZNEP) is scheduled to determine whether the user agrees with the proposed solution. To assist DFHZNEP, DFHZNAC sets flags, indicating the proposed action. These action flags are in DFHZNAC's transaction work area (TWA). In most cases, DFHZNEP can modify DFHZNAC's proposed actions. The only time that DFHZNAC overrides DFHZNEP's modification of the TWA is when a terminal is to be disconnected from CICS; that is, when DFHZNAC determines that the abnormal situation requires that CICS 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 DFHZNEP, DFHZNAC performs the actions specified in TWAOPTL (except when disconnecting terminals, as noted above), issuing messages and setting error codes, as necessary.

The CICS system has a pregenerated dummy DFHZNEP which simply returns control when DFHZNAC links to it. The system programmer needs to code DFHZNEP only if additional error processing beyond that performed by DFHZNAC is desired. DFHZNAC gives control to DFHZNEP 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 DFHZNEP has performed its functions, it returns control to DFHZNAC by issuing a DFHPC TYPE=RETURN macro instruction.

Upon entry to DFHZNEP, 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 DFHZNEP coding released with CICS. Linkage to DFHZNEP is provided by CICS.

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 DFHZNEP to determine any necessary actions. If DFHZNAC is scheduled because of loss of the connection between CICS and a logical unit, DFHZNAC abnormally terminates any transaction in progress at the time of the failure. DFHZNEP analysis and processing is permitted, but message retry should not be attempted.

The DFHZNAC error message is sent to the master-terminal log after linking to DFHZNEP. 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 WRITEQ TD command directly into DFHZNEP.

User Exit Routines for CICS DFHZCP

CICS 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 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.
- Before output data is transmitted (XZCOUT1).

Part 2. Design

This part deals with the components and functions that make up CICS. It consists of the following chapters:

- System Management Component
- System Services Component
- System Monitoring Component
- System Reliability Component
- System Support Component
- Application Services Component
- Intercommunication Facilities Component
- SNA Data Flow Control for Distributed Transaction Processing
- Intercommunication Using LU6.2.

Chapter 2.1. System Management Component

This chapter describes the major CICS management functions. The functions described are:

- Task management function
- Subtask management function
- Storage management function
- Program management function
- Table management function
- EXEC interface program
- User exit management function
- Task-related user exit management function
- Interval control function
- Terminal management function
- File management function
- DL/I data base support functions
- Transient data management function
- Temporary storage management function
- Journal management function
- Volume management function
- Sync point management function.

Task Management Function

The ability to process more than one task 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 (with a CICS-imposed maximum of 255). The number of active tasks is limited by the amount of available address space and the number of tasks specified as a maximum by the user.

Task Management Support for the DFHKC Macro

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 immediately to the requesting program, but may pass to the task dispatcher which selects a 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.

If a match is found in the PCT then task class maximum is verified and storage is obtained for the new dispatch control area (DCA). The dispatch priority is set in the DCA, and control returns to the caller. If the transaction is not valid, the

transaction CSAC is used instead to execute the abnormal condition program.

On first dispatching a task, task management acquires a TCA through storage management (described in "Storage Management Function" on page 47). 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.

If the task is marked in the PCT as a **primed** task, a primed storage area (PRA) is obtained. This area contains the TCA, and is also 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 canceled, 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 also results in storage still chained to the TCA being freed. (The DFHKC TYPE=DETACH macro is used only by CICS management modules.)

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 increases 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 decreased. 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. (The DFHKC TYPE=DEQALL macro is used only by CICS management modules.)

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 and so the event control block (ECB) is never added to the operating system wait list when CICS 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 chain to the suspend chain, if necessary in time-out 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. (The DFHKC TYPE=SUSPEND macro is used only by CICS management modules.)

Resume a Task (RESUME)

The task DCA is moved from the suspend 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. (The DFHKC TYPE=RESUME macro is used only by CICS management modules.)

Schedule a Resource (SCHEDULE)

SCHEDULE is used by interval control, transient data management, and BMS to tell task management that a task should be started. Task management cannot 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 macro.

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. (The DFHKC TYPE=AVAIL macro is used only by CICS management modules.)

Declare Task to Be Purgeable or Nonpurgeable (PURGE,NOPURGE)

The DFHKC TYPE=PURGE macro enables a macro-level application programmer to declare that a task may be purged if a system stall condition occurs (that is, when no task is able to continue, and no new task can be initiated). A similar macro (TYPE=NOPURGE) is available to **prevent** a task being purged in those conditions. The PURGE and NOPURGE macros override a transaction's purgeability status information as recorded (by the system programmer) in the program control table (PCT).

HPO Services

Services are included in task management for OS/VS2 versions of CICS for the high performance option (but these services are not part of the macro-level application programmer's interface).

- Attach the HPO transaction area (HTA)
- Detach the HTA.

Task Limits

Task control provides various tuning mechanisms. It sets an indicator **on** when the MXT limit has been reached. Other modules of CICS test the indicator before issuing a DFHKC ATTACH macro to start a new task. If the indicator is set, a module bypasses the ATTACH, and tries again later.

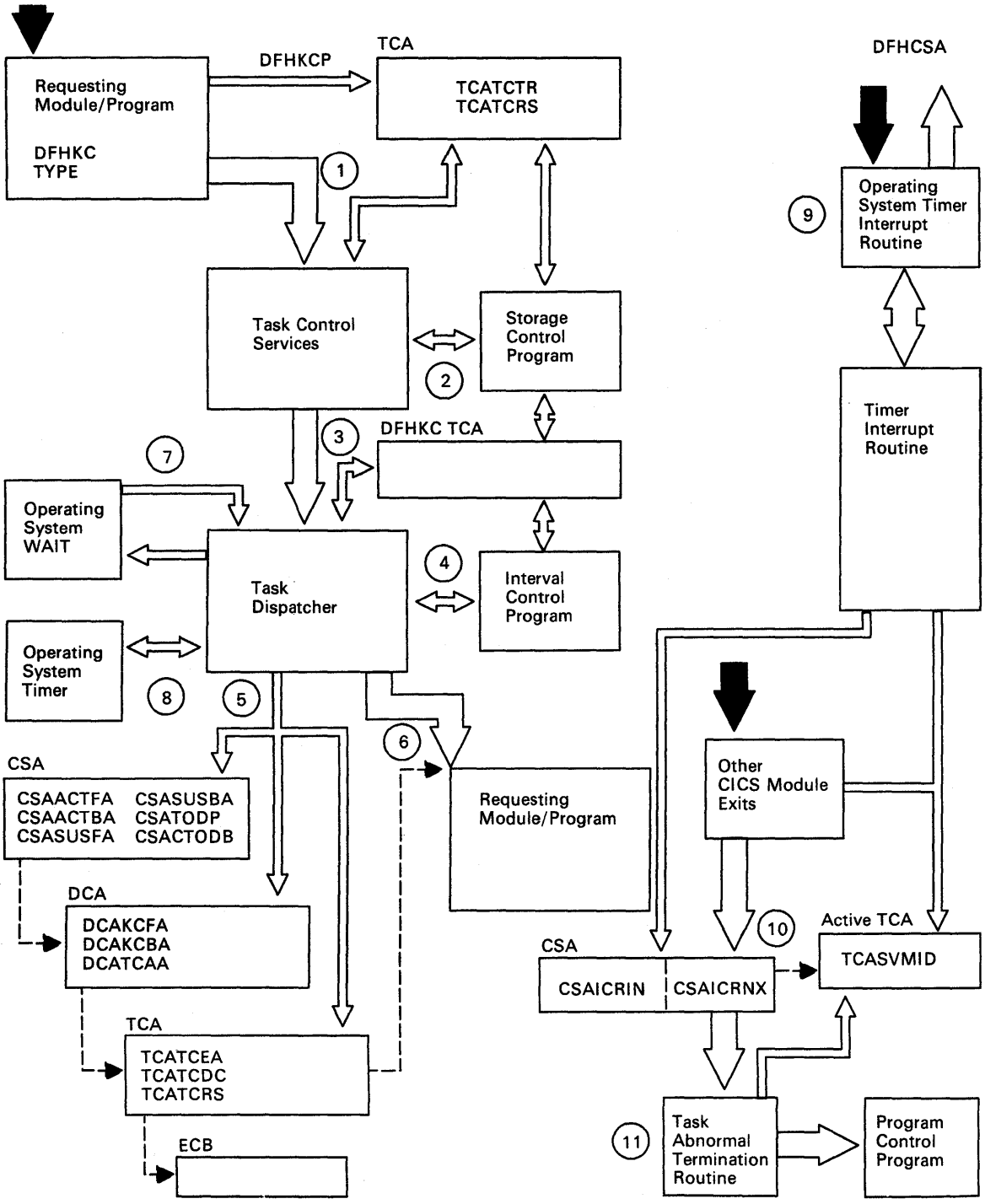
Task control enforces the AMXT limit by restricting the number of tasks on the active chain which it is prepared to consider when performing a dispatch scan.

Task control enforces the CMXT limit by refusing to dispatch a task if doing so would exceed the limit.

Task Management Modules (DFHKCP, DFHALP, DFHCSA, DFHHPSVC)

The task control program (DFHKCP) can be divided into three functional areas. The first of these supports the task control (DFHKC) macro instruction by performing or initiating a particular service requested, except for the SCHEDULE and AVAIL type of macros that are handled by DFHALP. As for other CICS service modules, the requesting task's TCA is the communication vehicle and is in control during the execution of the requested service. Unlike other CICS service modules, however, control is not always returned directly to the requesting program. Instead, control sometimes passes to the second functional area of task control, known as the task dispatcher. The CICS system function of selecting the task to be given control is performed. The third functional area comprises the routines that issue the operating system wait when CICS has nothing further to do.

Figure 7 on page 42 shows the relationships between the components of task management.



Notes:

1. *Issuing a DFHKC macro instruction causes a type of request code to be set in TCATCTR in the requesting task's TCA. Processing enters the task control macro instruction service logic. The register contents are saved in TCATCRS and in CSAOSRSA in the CSA for RESUME, UPDATE, and ATTACH HTA services, and the requested service is performed, or initiated. CSAOSRSA is also used because the routine that handles these requests is called from DFHKCP, and TCATCRS has already been used.*

   While processing the DFHKC macro request other modules may be called. DFHKCP also interfaces with DFHPCP to abend tasks in certain situations. During ATTACH macro processing, DFHTMP is used to find the PCT entry for the new transaction ID. DFHSPP is called during DETACH macro processing for DWE and sync point processing. If the task owned a terminal, then during DETACH macro processing DFHTCP/DFHZCP are called to free the terminal, DFHDIP is called to terminate outstanding requests and free the DIB, DFHICP is called to cancel any ICEs associated with the task and DFHALP is called to deal with AIDs.

2. *Storage control services are used by task control macro instruction service logic.*

3. *On entry to the task dispatcher, task control's TCA is activated as an intermodule communication vehicle.*

4. *If the expiry time of any ICE (describing a time-ordered event) has passed, the task dispatcher calls the interval control program 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 task dispatcher of task control. This value is used when operating system services are requested (see note 8).*

5. *Two task chains are maintained by the task control program: an active chain and a suspend chain. The task dispatcher scans the active chain when attempting to locate a dispatchable task. The suspend chain is scanned only during exception processing (for example, stall corrective action). The elements on the chains are dispatch control areas (DCAs), each of which contains pointers to the previous and next elements on the chain. In the case of the active chain, the pointers address the next higher priority DCA and next lower priority DCA (DCAKCFA and DCAKCBA, respectively). Each DCA also contains a pointer to the associated task's TCA (DCATCAA). The CSA contains pointers to the highest and lowest priority DCAs on the active chain (CSAACTBA and CSAACTFA, respectively). The dispatchability of a task is determined by examining the setting of the dispatch control indicator (DCATCDC) in the DCA. If it indicates that the task is waiting on the completion of some event, the ECB associated with the event (pointed to by DCATCEA) is tested for completion posting.*

   The suspend chain is not a priority-ordered chain. If a task has a time-out value associated with it, then task control will add the time-out task to the DCA chain in time-out sequence. CSASUSFA addresses the DCA with the temporarily most remote time-out value. KCP maintains an internal pointer (KCLTPTR) to the most imminent. Tasks with no time-out value are placed at the other end of the suspend chain (addressed by CSASUSBA). Placement in time-out sequence helps to minimize the overhead for tasks that have timed out.

6. *If the task dispatcher determines that a task is dispatchable, its TCA is activated, the task's register contents are restored, and control passes back to the module or program that requested the task control service. During this processing, operating system timer facilities may be used, as described in note 8, in support of the runaway task control facility of CICS.*

7. *If the task dispatcher finds no CICS task that is currently dispatchable, it releases control of the processor through operating system wait facilities, requesting that control be returned to CICS on completion of the next event. One of the events represents the expiration of a time interval as in note 8. Others may be associated with completion of I/O requests, and so on.*

8. *The task dispatcher interfaces with operating system timer facilities. When a task is running, timer interrupts are used to trap looping tasks as*

part of the runaway task process. When relinquishing control to the operating system, the timer is set to the time interval remaining until expiration of the next time-controlled event. This, in effect, requests that control be returned to CICS at a time no later than the expiration of that interval. The dispatcher continually updates the current time-of-day values in the CSA (CSATODP and CSACTODB), but does not use the timer facilities for this process.

9. *The task dispatcher either sets the operating system timer to a maximum time that CICS wishes to relinquish control to the operating system by a wait, or uses the timer as a means of interrupting an apparently looping (runaway) CICS task. When a time interval expires, the operating system gives control to the timer interrupt routine. Based on the setting of an indicator in the CSA (CSAICRIN), the timer interrupt routine determines which type of interval has expired. In a nonrunaway case, the timer interrupt routine merely posts the CICS timer ECB (CSATTECB) and returns control to the operating system.*

10. *If a runaway task interval has expired, the timer interrupt routine in DFHCSA takes steps to purge (abnormally terminate) the apparently looping task. The setting of bits in a system-maintained field in a task's TCA (TCASVMID) indicate whether a task is currently executing in a CICS management module or the application program itself. In the latter case, the timer interrupt routine ensures that an immediate purge will be initiated when CICS is again given control. This is done by issuing a system abend, and nominating task purge as the recovery routine in the ESTAE exit. If the runaway task interval expired while a CICS management module was executing, the abnormal termination is deferred until control returns to the application program. The timer interrupt routine then returns control to the operating system.*

11. *The linkage in the CSA causes entry into the task abnormal termination routine, which requests abnormal termination by means of a program control DFHPC TYPE=ABEND macro instruction. The CICS program control program terminates the task.*

Tasks on the active chain may be running under SRBs, rather than the CICS TCB, during execution of certain CICS services. Such tasks must not be dispatched by the normal (TCB mode) CICS task dispatcher. Upon completion of the SRB mode service, a task becomes eligible for normal TCB mode dispatch again. To indicate that a task has become eligible in this way, the SRB mode service posts an ECB (DCAPECB in the task's DCA), and places the HTA of the task on the stage chain (SRASTGCH). The task dispatcher dispatches from the stage chain in preference to the active chain except when the "next dispatch time" for the terminal control task (CSATCNDT) expires.

Subtask Management Function

Some synchronous operating system requests issued by CICS modules could cause CICS to be suspended until the requests had completed. To avoid the resulting response-time degradation, certain requests are processed by the general-purpose subtask management program, DFHSKP. A CICS module calls DFHSKP to execute a routine within the module under an operating-system subtask.

DFHSKP does the following:

- Schedules a subtask to execute a routine (called an SK exit routine).
- Allows an SK exit routine to wait on an operating system event control block (ECB).
- Manages subtask creation, execution, and termination.
- Handles program checks or abends within the SK exit routine.

DFHSKP consists of the programs DFHSKM, DFHSKC, and DFHSKE.

Subtask Management Modules (DFHSKM, DFHSKC, DFHSKE)

DFHSKM

A DFHSK macro invokes DFHSKM to cause a routine to be executed under an operating system subtask. DFHSKM chooses a subtask to execute the request unless the caller has specified a particular subtask.

DFHSKM determines if the subtask is dead, not started, or running. The subtask is called dead if it has terminated itself, or could not be attached. If the subtask is dead and the user coded SYNC=YES in the DFHSK macro, the request is processed synchronously. Synchronously means that DFHSKM executes the request under the CICS task control block (TCB).

If the subtask has not started, then DFHSKM attaches a CICS task specifying the entry point of DFHSKC to execute. DFHSKM then waits on an ECB in the subtask control area (SKA) for the subtask and continues when the ECB is posted by DFHSKC, indicating that the subtask has been initialized.

DFHSKM then creates a work queue element (WQE) which represents the work to be performed under a subtask. The WQE is added to the work queue for the subtask. When the work ECB of the subtask is posted, signaling work to do, DFHSKM issues a wait on the work-complete ECB in the WQE. This ECB is posted when the WQE has been processed by the subtask. DFHSKM returns control to the caller, indicating the outcome of the processing.

Should the subtask processing the WQE fail before completion, DFHSKM is informed and attempts to execute the request synchronously if the caller so specified.

When CICS terminates, it issues a DFHSK CTYPE=TERMINATE macro to terminate the subtasking mechanism. DFHSKM sets a flag in each subtask control area (in DFHSKP static storage) indicating that the subtask should terminate. DFHSKM then posts the subtask work ECB to signal the subtask to examine this flag.

DFHSKM is also invoked by deferred work element (DWE) processing and by the DFHKCP task cancel.

When DFHSKM decides to process a WQE synchronously, control is passed to the routine specified by the caller. This routine may not complete normally and, so that DFHSKM does not lose the WQE because the task abended, it creates a DWE containing the address of the WQE. Should the task be abended, the DWE processor adds the WQE to the free queue.

The master terminal operator may attempt to cancel a CICS task while DFHSKM is executing and the WQE is in use by the subtask. If task cancel is allowed to proceed, the WQE could be reused, and corrupted by the subtask which has addressability to it. In this case, task cancel is intercepted and delayed while DFHSKM is waiting for the subtask to process the request. When WQE processing is complete, the invoker of DFHSKM is informed, by a return code, that the task was canceled.

DFHSKC

DFHSKM invokes DFHSKC using the DFHKCP attach logic to start an operating system subtask and wait for its completion. DFHSKM passes the address of the subtask control area in the facility control area address (TCAFCAAA) in the TCA.

DFHSKC issues a CICS GETMAIN for shared storage to pass to the subtask for use as its automatic storage. The length required is in a field in DFHSKE containing the automatic storage requirements. DFHSKC issues the ATTACH macro with the ECB option to attach the operating system subtask and passes the address of the subtask control area.

DFHSKC issues the CICS SVC to authorize the TCB of the subtask to use the SVC.

DFHSKC issues a KC wait on the attach ECB. The module is suspended until subtask termination, when the ECB is posted. On termination, the subtask puts a return code in the subtask control area.

When the subtask completes, DFHSKC cleans up the subtask work queue. It then frees the automatic storage and returns to DFHKCP.

DFHSKC writes messages to CSMT from this module if it was unable to attach an operating system subtask or the subtask indicated that its termination was not normal.

DFHSKE

When the subtask manager DFHSKM, executing on behalf of a CICS task, decides that a subtask is to be started, it attaches a CICS task using the DFHKC ATTACH macro and specifying the entry point of DFHSKCNA. This CICS task attaches the subtask and waits for subtask completion by means of the ECB parameter coded in the ATTACH macro.

The ATTACH macro specifies an entry point in DFHSIP (known to MVS by an IDENTIFY macro issued in DFHSIP). DFHSIP then branches to the entry point of DFHSKE, whose address is in the subtask control area.

Note: DFHSIP remains in storage after initialization has completed.

The subtask reverses the order of the in-progress queue to service requests on a first-come, first-served basis. It the loops round the in-progress queue and, for each WQE, branches to the program specified in the WQE (the SK exit routine).

The exit routine returns control to DFHSKE, either indicating that the exit routine has completed by issuing a DFHSK CTYPE=RETURN macro or requesting that execution of the SK exit is suspended until an ECB specified by the exit is posted by some operating-system component.

When a return is requested, the ECB in the WQE is posted, causing DFHKCP to resume the CICS task that was waiting for the SK exit to be complete. When a WAIT is requested, the WQE is added to the waiting queue which is processed later.

When all WQEs in the in-progress queue have been processed, DFHSKE then examines the waiting queue. If any WQEs are on this queue, their ECB addresses are inserted into an operating-system multiple-wait queue. The subtask work ECB (posted when a WQE is added to the work queue) is put at the top of this multiple-wait list. An operating-system multiple-wait is then issued.

When the subtask regains control, an ECB has been posted. This can be either because more work has arrived or an ECB belonging to an exit routine has been posted.

The WQEs on the waiting queue are scanned, and those whose ECB has been posted are moved to the in-progress queue, with a flag on indicating that an SK exit routine is to be resumed.

Control returns to the beginning of this program which examines the work queue and proceeds as described earlier.

DFHSKE handles program checks and operating system abends. If an abend exit is driven when processing a WQE, then that WQE is blamed and processing of it terminates. The CICS task requesting the processing is informed of the problem.

If an abend exit is driven when a WQE is not being processed, then it is assumed to be a problem in the subtasking program. The abend is handled, and a count of failures is increased. When a threshold is reached, the subtask terminates.

The MVS exits are ESTAE and SPIE.

For normal termination, DFHSKE loops, processing WQEs and waiting when there is no work to do. The subtask checks a flag in the subtask control area to see if it has been requested to terminate. If the flag is set, the subtask terminates, indicating normal termination by setting a response code in the subtask control area for the attacher, DFHSKC.

Abnormal termination may occur when the error threshold has been reached. The subtask terminates, but sets an error return code in the subtask control area for the attacher to see. The attacher, DFHSKC, then cleans up any outstanding WQEs on the subtask queues.

Storage Management Function

All dynamic storage for CICS and for the user-written application programs is controlled by the storage management module DFHSCP. Requests to acquire or release dynamic storage are communicated to storage management by CICS macro instructions. CICS 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 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 verifies all requests for dynamic storage. This helps in isolating and terminating transactions that inadvertently destroy storage chains. If storage chains are destroyed, the storage control recovery program (DFHSCR) attempts to reconstruct them so that dynamic storage remains usable by CICS 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 running.

System Overload Detection

To relieve an overload, CICS uses a technique that involves a storage cushion. The cushion is a quantity of dynamic storage held in reserve by CICS. When a request for dynamic storage cannot be satisfied, the cushion is released for use by current tasks, the initiation of new tasks is inhibited, and the message DFH0506 is issued. When the demands for dynamic storage have diminished, the cushion is reacquired by CICS, new transactions can be initiated, and the message DFH0507 is issued.

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
- The number of times requests for acquisition are satisfied from above the 16 megabytes line.
- 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 function:

Storage Acquisition
Allocates dynamic storage for 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.

CICS with MVS/XA

For MVS/XA, the storage control program also allocates and frees storage from above the 16 megabytes line using the MVS/XA GETMAIN and FREEMAIN macros. The storage areas chained from CSASCXCA are obtained from MVS subpool 1 and the storage areas chained from TCASCXCA are obtained from subpool 0.

Storage Management Module (DFHSCP)

Figure 8 on page 49 shows the relationships between the components of storage management.

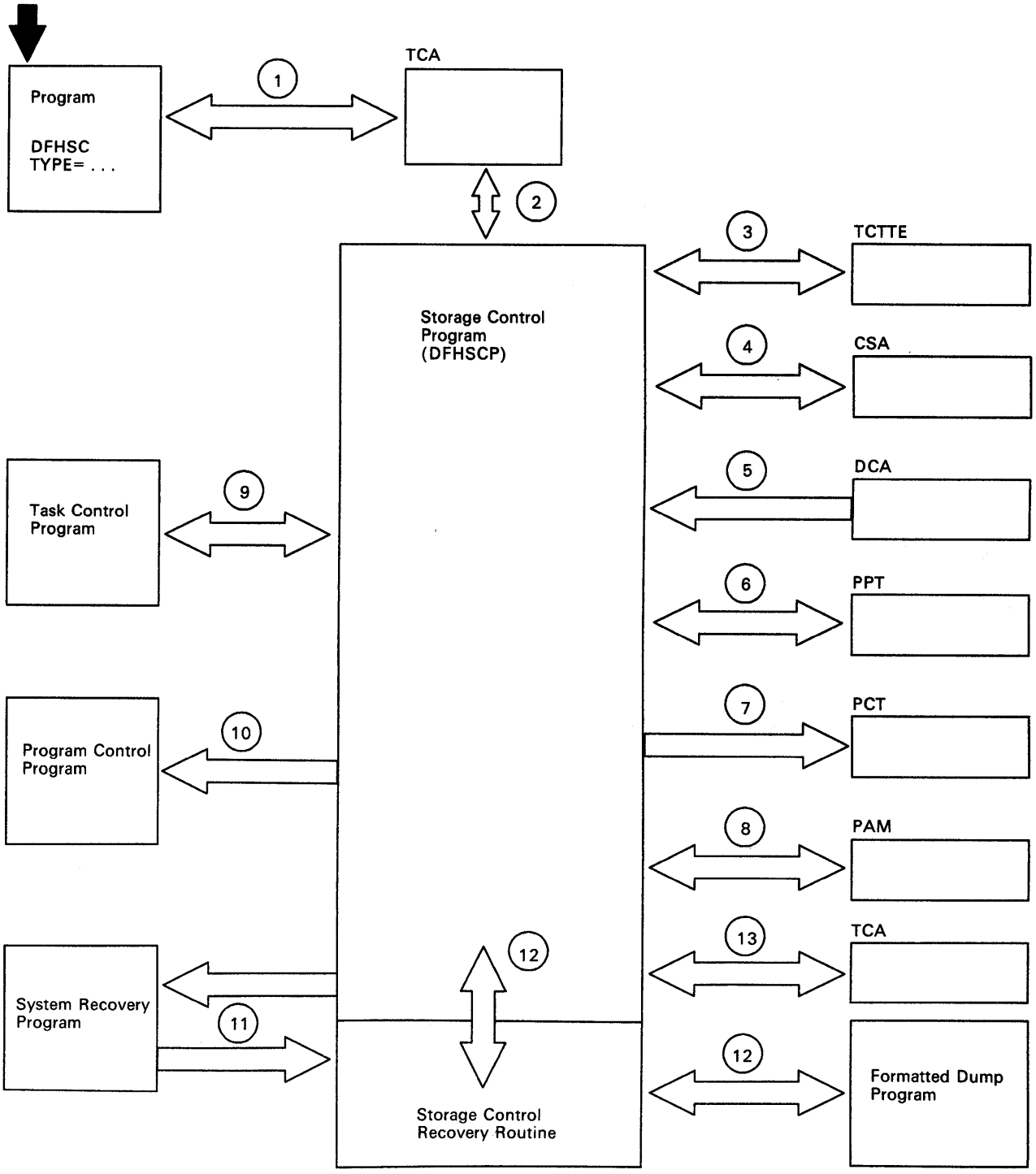


Figure 8. Storage Control Interfaces

Notes:

1. *All CICS and user-written programs communicate their requests for storage control services through the TCA. The address of acquired storage is returned to the requesting program in the TCA.*

2. *Storage control determines the type of request by referring to the TCA; it then 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 chained off the TCA can be frozen to aid problem determination (for more information, refer to the CICS/MVS Problem Determination Guide). Requests to free frozen storage are not honored. The storage is only freed when the transaction that acquired it is terminated.*

3. *Storage control 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. Terminal storage chained off the TCTTE can also be frozen, as mentioned in note 2.*

4. *Storage control 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.*

5. *The chain of suspended DCAs is searched by storage control when an SOS condition occurs. Suspended requests for storage are retried whenever storage becomes available for allocation.*

6. *When the system is approaching an SOS condition, storage control searches the processing program table (PPT) for programs residing in dynamic storage that are not currently in use. Program storage for programs not in use is freed, and the PPT is changed to reflect the fact that the program is no longer in storage.*

7. *A count of the number of storage violations associated with a transaction identification is kept in the program control table (PCT).*

8. *The page allocation map (PAM) 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.*

9. *If an unconditional request for storage cannot be satisfied, storage control issues a DFHKC TYPE=SUSPEND macro instruction, which is a task control macro instruction used only by CICS 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 management modules) is issued to request task control to start the task.*

10. *If an invalid request is issued, or an address specified in a request is invalid, storage control issues a program control DFHPC TYPE=ABEND macro instruction to request the program control program to terminate the task.*

11. *If the storage control program program-checks 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.*

12. *If storage control detects a storage violation while servicing a DFHSC TYPE=GETMAIN or DFHSC TYPE=FREEMAIN request and storage recovery is generated as part of the CICS system, control is passed to the storage control recovery routine. The storage control recovery routine optionally generates a formatted dump, then tries to fix the storage problem. If the storage can be repaired, the current task is abended; if not, CICS is abended.*

13. *Storage items with task-related lifetime are normally put on a chain anchored in TCASCCA, so that they can be freed automatically at task termination or abend. If the task is designated as* **primed** *in the PCT, the task-related areas are allocated contiguously from within the primed storage area (PRA). A PRA is obtained with the TCA and other system control blocks at the start of a task, and freed with them at task end.*

Program Management Function

Program management controls and supervises CICS application programs. Program management macro instructions are used to load programs, link or transfer control to programs, retry a program (as part of transaction restart), 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 library in relocatable format and are accessed through the CICS asynchronous program fetch data facilities. This loading facility allows CICS execution to continue during program load time. The real-time relocatable program library is a standard partitioned data set. Programs are prepared for this library by the OS/VS linkage editor. 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 Performed without Communication with the Application Program

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 services to save information, passes control to the appropriate CICS management function, and subsequently returns to the HLL program.

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

Asynchronous Program Fetch: Fetches application programs from the direct-access storage device into dynamic storage while allowing other transaction activity to proceed during the I/O operation. In the MVS/XA environment, this function is only provided for programs link-edited with the option RMODE = 24. Programs link-edited with RMODE = 31, or RMODE = ANY, are loaded synchronously above the 16-megabyte line using the OS loader.

Program Management Services Performed in Response to Requests from an Application Program (or Another CICS Function)

The following services are performed by program management in response to a specific request from either a user application program or another CICS 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, 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 CEMT and CSMT NEWCOPY function by obtaining the information about the new program.

Program Management Module (DFHPCP)

Figure 9 shows the relationships between the components of program management.

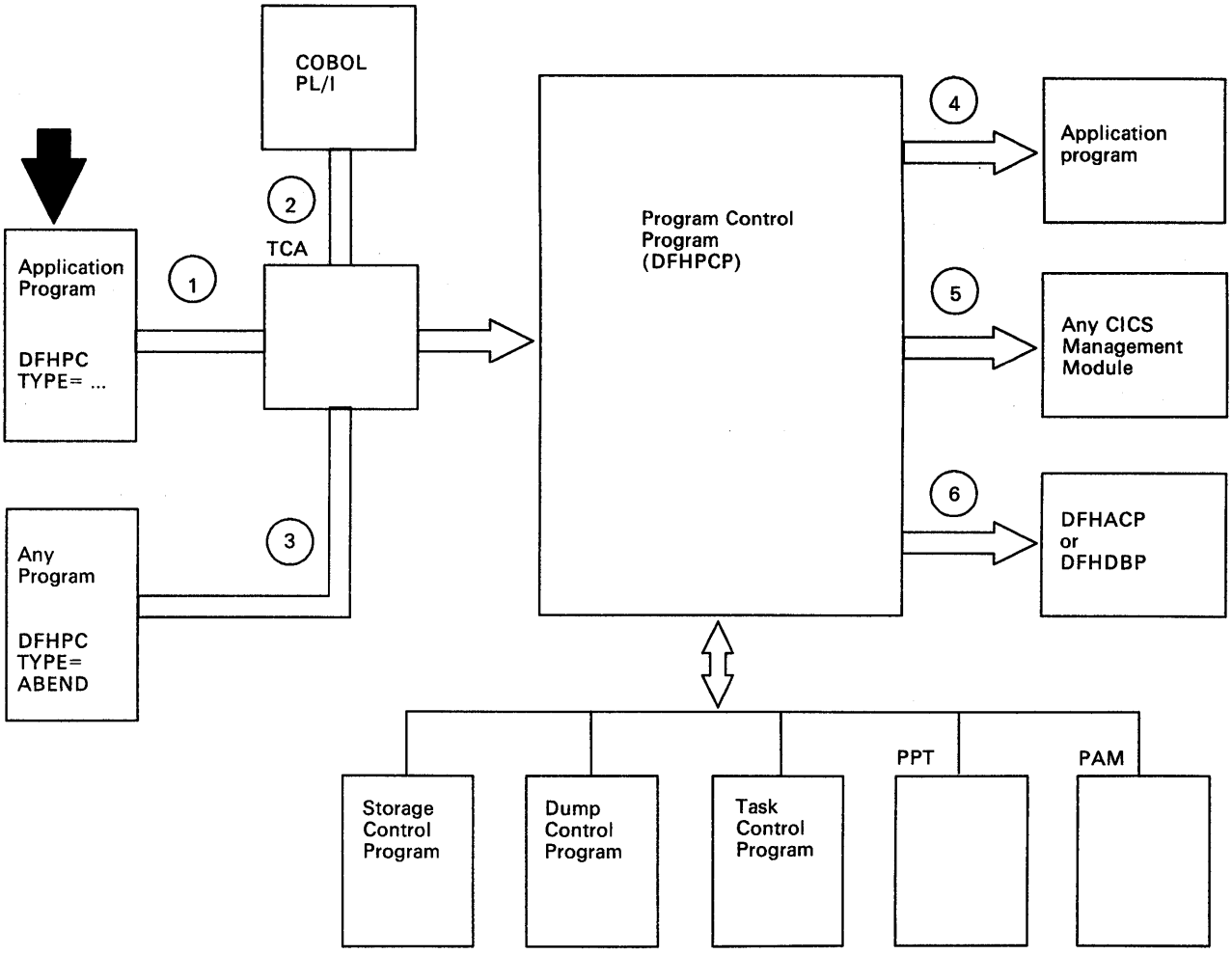


Figure 9. Program Control Interfaces

Notes:

1. *Program control (DFHPC) macro instructions establish fields in the TCA to communicate with the program control program.*

2. *COBOL and PL/I modules that use the macro interface establish fields in the TCA and communicate with program control on the way to other required CICS functions.*

3. *Any CICS or user-written module may issue a program control DFHPC TYPE=ABEND macro instruction to request that a task be abnormally terminated.*

4. *Control is passed to the calling program or the called program, on the successful completion of a LINK, XCTL, LOAD, SETXIT, or RETURN routine.*

   If the called program uses the command interface, DFHPCP invokes DFHEIP to initialize the command interface environment.

5. *Calls from COBOL and PL/I modules using the macro interface may result in an exit to any CICS management function.*

6. *In the event of an ABEND, control may be passed to a user-specified SETXIT module or subroutine. If, however, there is no SETXIT or the SETXIT routine(s) returns abnormally, the ABEND continues. In this situation DFHACP will be invoked unless dynamic backout is required, in which case DFHDBP will be invoked.*

Table Management Function

Locating, adding, and deleting entries in the PCT, PPT, and TCT, and locating entries in the DCT, FCT, and TCT are performed by the table management program (DFHTMP). Entries in these tables are also called resources. Because the structures of tables vary as entries are added or deleted, and a quick random access is required, a hash table mechanism is used to reference the table entries.

Hash Table

The hash table is a set of pointers that are the addresses of directory elements of table entries. A directory element is a set of pointers; one of these pointers is the address of the table entry, the remaining pointers are the addresses of the next elements of various chains used in the different operations of table management. An example of a hash table is shown in Figure 10 on page 55.

Table management logically combines the characters of the name of the resource, and transforms the result to give an integer that is evenly distributed over the hash table size.

When an entry is located or added, table management places it at the head of its chain. Thus frequently used entries tend to have the minimum search times.

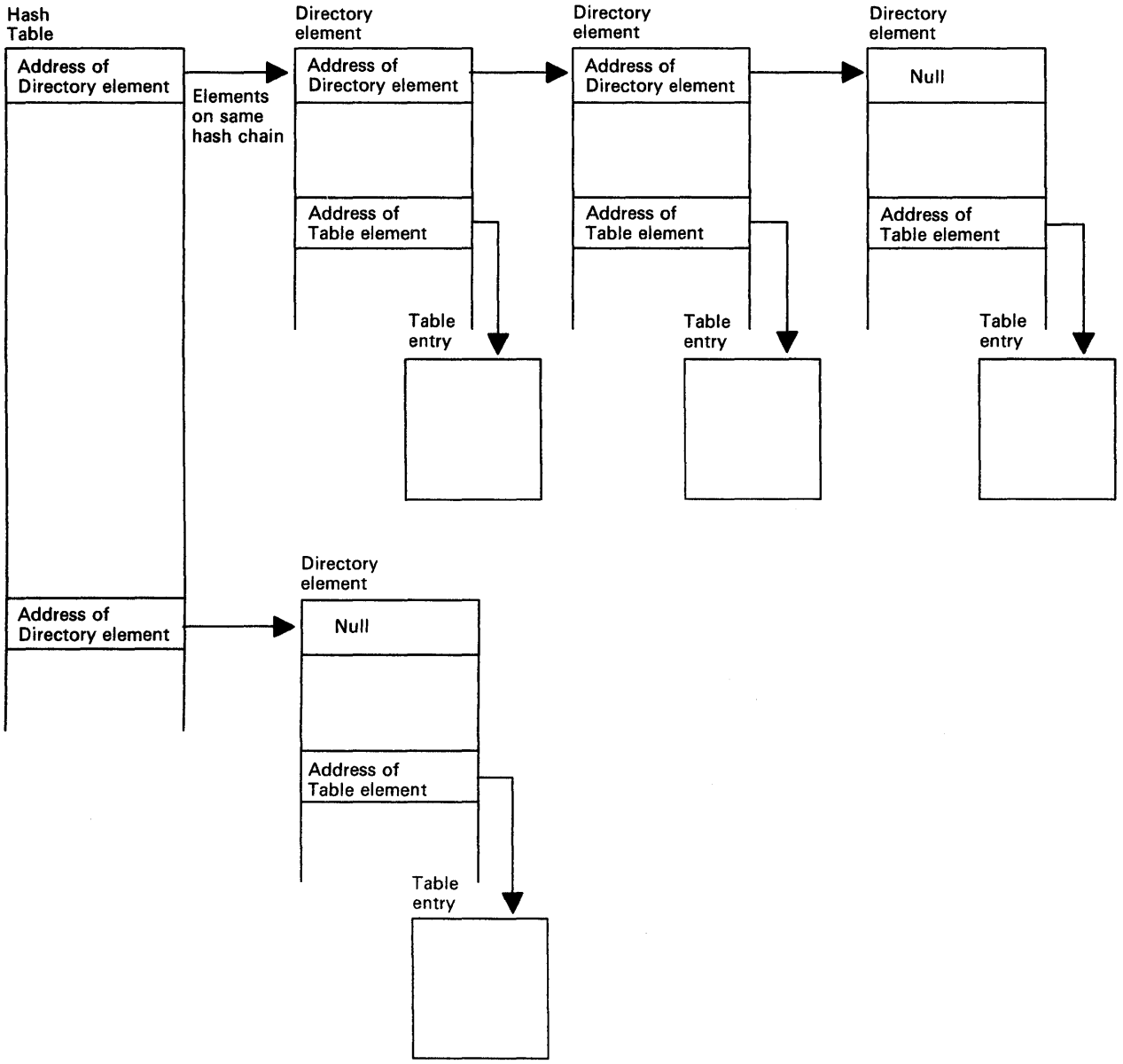


Figure 10. Hash Table for Table Management

If the hash chains become very long, table management creates a larger hash table if storage is available. The hash table is enqueued before, and dequeued after the reorganization, so that no references to the table can be made during reorganization.

For the names of resources that are in remote systems, a separate hash table, called the secondary index, is created.

Functions of Table Management

Table management performs the following functions:

- Locating a table entry – given a name, finding the address of the table entry.
- Obtaining the next table entry – given a name, finding the address of the table entry in collating sequence order.

- Obtaining the next alias name – given a name, finding the alias name (if any).
- Adding a table entry – given a table entry, adding it into the table
- Creating an alias – given a name, assigning an alias name.
- Adding a secondary index entry – given a primary name and a secondary name, adding the names to the secondary index.
- Deleting a table entry – given a name, deleting it and any associated alias.
- Locking a table entry – given a name, assigning a read lock to it.
- Unlocking a directory entry – given a name, removing the associated read lock.
- Quiescing a table entry – given a name, marking it as busy.
- Transferring a lock to a different task – given a name and the address of a target TCA, transferring the read lock to the target task.
- Creating an index – creating a hash table of a given type.

Read Locks

Read locks are used to prevent a table entry being deleted by table management.

A read lock is a fullword of storage. When DFHKCP attaches a task, it allocates storage for a number of local read locks; this storage is pointed to by TCATMLBA in the TCA. Local read locks are not acquired for table entries that cannot be deleted.

Global read locks are used by the CICS modules that execute independently of any task. They reside in the table management static storage area that is pointed to by the location SSATMP in the CSA.

EXEC Interface (DFHEIP)

The EXEC interface program is used by application programs using the command-level interface. It invokes the required CICS services, or DL/I services, on behalf of the application program.

The EXEC interface program is divided into a number of separate modules. It consists of the EXEC interface program nucleus (DFHEIP) and one module for each of the CICS management modules. These separate modules are called the EXEC interface stubs and are named with the following convention:

The stub for DFHFCP is DFHEFC,

the stub for DFHTDP is DFHETD,

and so on.

The stub for DL/I services is DFHERM. The management module for DL/I is DFHEDP.

Figure 11 on page 57 shows the relationships between the components of EXEC interface program processing.

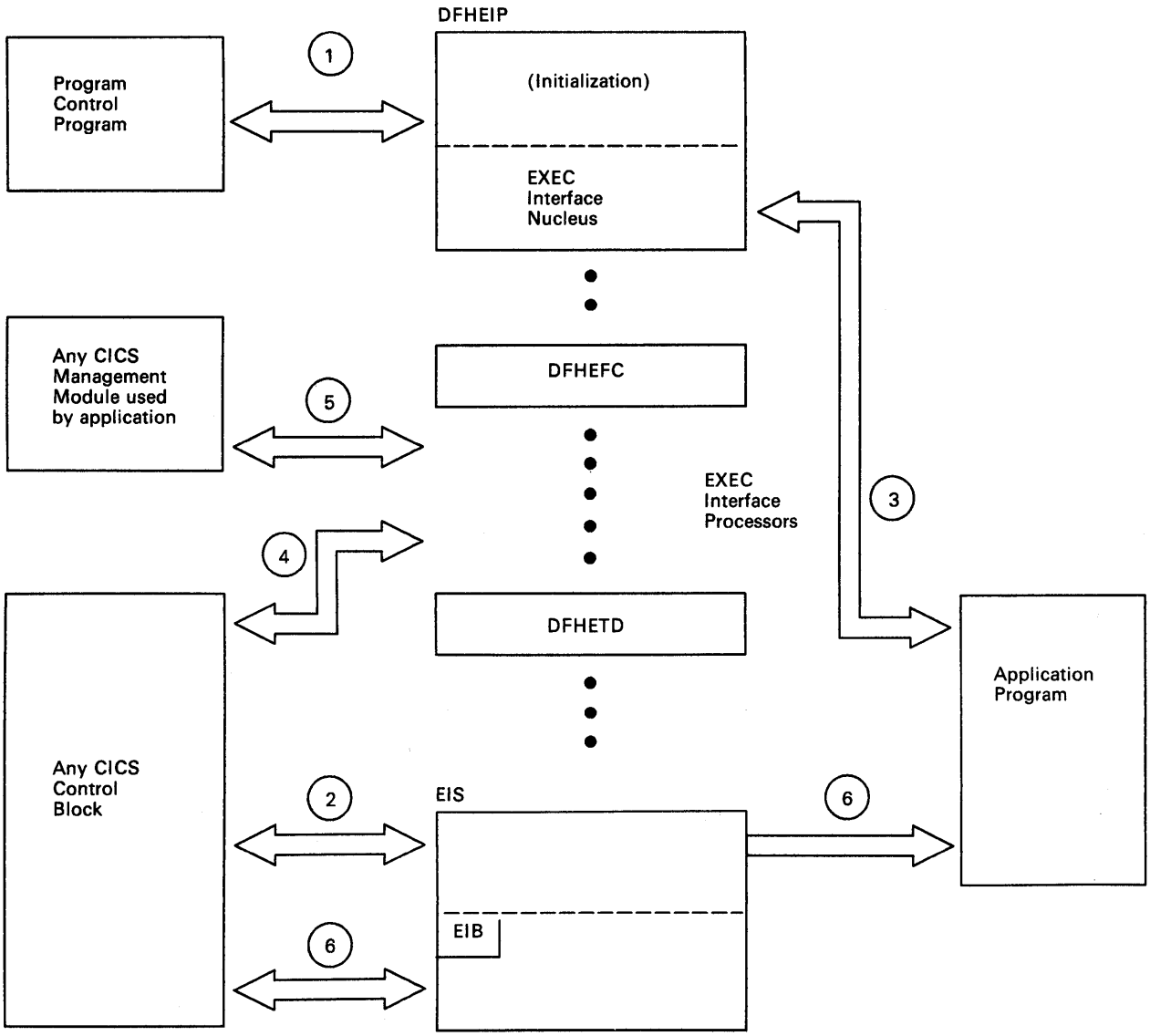


Figure 11. EXEC Interface Program Interfaces

Notes:

1. *When DFHPCP loads an application program that uses the command-level interface, it invokes the DFHEIP initialization routine to set up the command-level interface environment. DFHEIP then invokes the application.*

2. *The initialization routine obtains and initializes the EXEC interface structure (EIS). This contains an EXEC interface block (EIB) which is used to pass information to the application program.*

3. *The statements inserted by the command-language translator for each EXEC CICS command include a CALL statement that causes the application to invoke DFHEIP. The nucleus module DFHEIP then invokes the appropriate EXEC interface stub.*

4. *The arguments passed by the application are moved into CICS control blocks by the appropriate EXEC interface stub.*

5. *The CICS management module is invoked by the corresponding EXEC interface stub.*

6. *Information is returned to the application program by DFHEIP through EIB.*

User Exit Management Function

User exit management provides an interface that allows the user to run exit programs at selected points (known as exits) in CICS management modules. The exit programs are separate from the management modules and are associated with them dynamically by means of the EXEC CICS ENABLE command (see the *CICS/MVS Customization Guide* for a description of how to use exit programs).

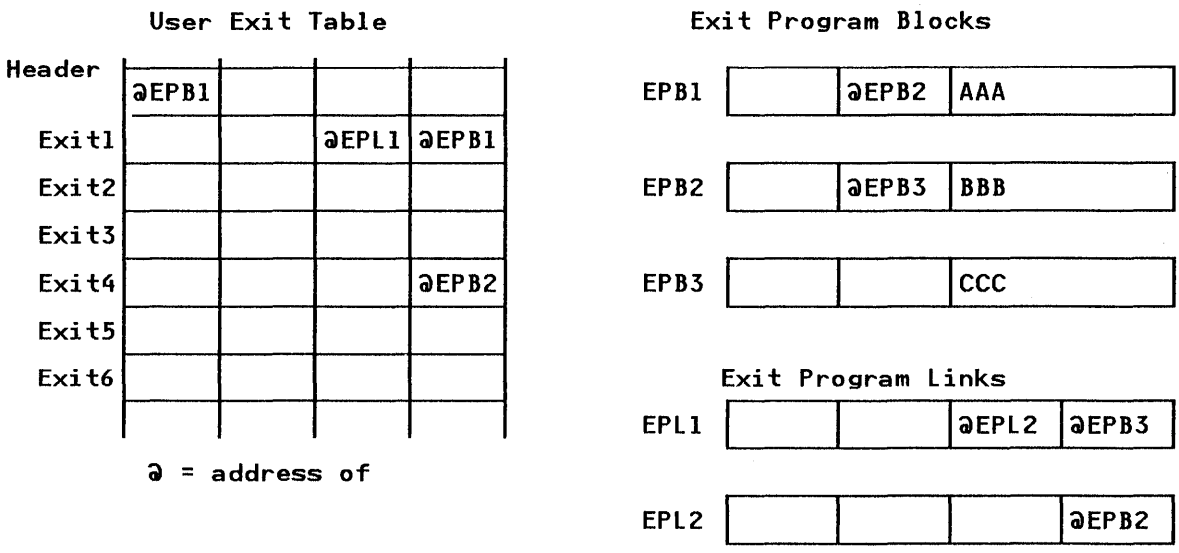
An exit can have more than one exit program and an exit program can be shared by more than one exit. Work areas can be set up for the exit programs, and several exit programs can share a work area. For some exits, the continuation of the management module can be controlled by a return code.

Each exit is identified internally by an exit-number. The user exit table (UET), contains a UET header and an entry for each exit, in exit-number order. The UET is addressed from CSAUETBA in the CSA and exists throughout the life of CICS.

Each enabled exit program is represented by an exit program block (EPB). This exists only while an exit program is enabled. The EPBs are chained together in order of enablement. The UET header points to the first EPB.

Each activation of an exit program for a particular exit is represented by an exit program link (EPL) which points to the EPB for the exit program. The first EPL for each exit is contained in the UET entry. If an exit has more than one exit program, then additional EPLs are obtained to represent each subsequent activation. These additional EPLs are chained off the UET entry in order of activation. Thus, for each exit, its EPL chain defines the exit programs that are to be executed at that exit, and the order of execution.

The user exit interface (UEI) control blocks are illustrated in Figure 12 on page 59.



Notes:

1. *There are three enabled programs, AAA, BBB, and CCC.*
2. *Exits 1 and 4 are associated with these exit programs.*
3. *Exit programs AAA, CCC, and BBB have been activated for exit 1.*
4. *Exit program BBB has been activated for exit 4.*

Figure 12. UEI Control Blocks

User Exit Management Modules

User Exit Handler

At each exit in a management module there is a branch to the user exit handler DFHUEH. This module scans the EPL chain for that exit, and invokes each started exit program in the chain, passing it a parameter list, and a register save area. On return from each exit program, the return code is checked, and a current-return-code (maintained by DFHUEH for return to the management module) is set, as appropriate.

User Exit Manager

The user exit manager DFHUEM processes EXEC commands which are entered by an application program or the command interpreter to control user exit activity. DFHUEM contains three routines, corresponding to the three commands, as follows:

1. ENABLE checks if an EPB already exists for the exit program specified in the PROGRAM operand.

   If an EPB is not found, and the ENTRY operand is not specified, the exit program is loaded. A new EPB is obtained and added to the chain. The name and entry address of the exit program is placed in the EPB. If the GALENGTH operand is specified, a work area is obtained and its address and length placed in the EPB. If the GAPROGRAM operand is specified, the address of the EPB for the exit program specified in the GAPROGRAM operand is placed in the new EPB.

   If the EXIT operand is specified, the EPL chain for the specified exit-number is found. A new EPL is obtained, if necessary, and added to the chain. The address of the EPB for the exit program specified in the PROGRAM operand is placed in the EPB. The activation count in the EPB is increased by 1.

   If the START operand is specified, the start-flag in the EPB is set on.

2. DISABLE finds the EPB for the exit program specified in the PROGRAM operand.

   If the STOP or EXITALL operand is specified, the start-flag in the EPB is set off.

   If the EXIT operand is specified, the EPL chain for the specified exitid is found. The EPL pointing to the EPB for the exit program specified in the PROGRAM operand is removed from the chain and the activation count reduced by 1.

   If the EXITALL operand is specified, all EPL chains are scanned and, for all EPLs pointing to the EPB for the exit program specified in the PROGRAM operand, the EPL is removed from its chain. The exit program is deleted unless the ENTRY operand was specified when it was enabled. The EPB is removed from the chain. Any work area used by the exit program is released unless it is still being used by another exit program. Any EPB that is no longer required is also released.

3. EXTRACT-EXIT finds the EPB for the exit program specified in the PROGRAM operand. The work area's address and length are extracted from this EPB (or from the EPB that owns the work area) and placed in the user's fields specified in the GASET and GALENGTH operands.

Task-Related User Exit Management

The task-related user exit interface is comparable with the EXEC interface. When an application program requests the services of a resource manager, it does so by a module called the task-related user exit. The exit receives arguments from the application program, and passes them on to the resource manager in a suitable form.

The advantage of this method is that if the resource manager is changed, the application program that invokes the resource manager should not need to be changed too.

The exit is part of the resource manager program(s). The name of the exit, or the name of the entry to the exit, is specified by the resource manager, and each application program that invokes the resource manager has to be link-edited with an application program stub that refers to that name.

The exit is enabled and disabled using the user exit manager. For enabling, the resource manager can specify the size of a transaction-related work area that it requires.

The exit, when enabled and subsequently driven, receives arguments in the form specified by the DFHUEXIT TYPE=RM parameter list (see the *CICS/MVS Customization Guide*, or the *CICS/MVS Data Areas*). Register 1 points to this parameter list. Register 13 points to the address of a save area, rather than the address of the CSA. The save area is 18 words long, with registers 14 through 12 stored in the fourth word onward.

Responses to the request are indicated by bit settings in the schedule flag word pointed to by UEPFLAGS, and also by means that are specific to the architecture of the application interface, for example by moving data into storage areas passed by the call or into the caller's register 15.

The main control block used by the task-related interface is the transaction interface element (TIE). A TIE is created by DFHEIP on the first call by a transaction to each resource manager, and it is chained to the TCA for that transaction. Part of the TIE is a DWE that is the interface between DFHERM and the sync point program. Consequently, the TIE is also part of the DWE chain for that TCA.

Task-Related User Exit Implementation

The state of an exit is managed by DFHUEM, which is described under "User Exit Management Modules" on page 59. For an exit, the TALENGTH argument and a profile in the form of a bit-string are held in the exit program block (EPB). These arguments are not processed until the occurrence of an application program CALL that explicitly names the exit.

Entry to the exit is through the task-related user exit interface, which comprises:

An application stub provided with the exit, but generated using the CICS-provided macro

DFHRMCAL. It is this stub which explicitly names the exit, and which is link-edited with each application program that uses the application program interface (API) of the resource manager.

DFHEIP, which is entered at DFHEIPCN by the application stub, in much the same way as EXEC CICS commands are routed at execution time.

DFHERM, which receives control when DFHEIP discovers that the call is not for a CICS management function, but for a named exit.

DFHERM receives a set of registers (those of the caller, for example, the application program), and a routing argument which names the exit. This routing argument is constructed by DFHRMCAL, in the application stub, and is not normally visible to the application programmer. DFHERM retrieves the name of the requested exit from the routing argument, and scans any existing transaction interface elements (TIEs) that are chained from the task's TCA looking for a TIE associated with the named exit. If such a TIE is not found, it searches the installed exits on a chain of EPBs, looking for the matching name. On finding a match, DFHERM constructs a TIE to represent the connection between that task and the exit. The TIE is initialized from information provided in the EPB; the TALENGTH argument defines the size of a task-local work area which can be thought of as a logical extension of the TIE (a kind of TWA that is exclusive to that exit). The profile string is also copied into the TIE.

Having created, or found, the appropriate TIE, DFHERM stacks (stores in a last-in, first-out manner) various parts of the program execution environment – the status of HANDLE commands, file browse cursors, the EXEC interface block (EIB), and so on – and builds a parameter structure which is essentially a superset of that built by DFHUEH. Additional arguments include the task-local work area, the profile referred to above, and an 8-byte identifier of the current unit of recovery (LUW), and so on.

DFHERM then passes control to the exit's entry point using standard CALL conventions, in which register 13 addresses a save area for DFHERM's own registers, register 14 addresses DFHERM's next sequential instruction, and register 1 addresses the passed parameters. This is a vector of addresses which include that of the caller's register save area. Any changes the exit makes to arguments of the application program interface (API), or to the contents of the caller's register save area, are not examined by DFHERM when it regains control, since they are not part of the CICS task-related user exit interface – rather they are the concern of the caller and the exit. However, the exit can request DFHERM to schedule certain actions by means of the profile argument. For example, the exit can request that it be informed (driven) when commitment of resources (sync-pointing) is taking place, or the exit can request that DFHERM no longer routes API calls to it from this task.

Finally, on regaining control from the exit, DFHERM unstacks the objects that it had previously stacked, and returns to the caller. The state of the cursors, HANDLE labels, etc., is apparently unchanged by the actions of DFHERM or the exit. Note that the exit may have used EXEC CICS HANDLE commands; this does not interfere with the caller's HANDLE status.

In the discussion of DFHERM so far, the term caller has been thought of as the application program. However, a caller can be one of several potential invokers, that is functions such as the sync point manager (DFHSPP), the task manager (DFHKCP/DFHZSUP), or the CICS monitoring facility (DFHCMP). The exit can set appropriate bits in the profile so that if the corresponding function is subsequently invoked, it in turn will call the exit. The exit can determine the identity of the caller from the first argument. This argument, passed by DFHERM, always has its first byte equal to X'00'. (If the first byte is other than X'00', then the exit has been entered from DFHUEH as a global user exit.) DFHERM sets the second byte of this argument according to the type of caller, thus indicating which interface is addressed by the caller's register save area. The second byte is X'02' for an application program, X'04' for the sync point manager, X'06' for the CICS monitoring facility, and X'08' for the task manager. Any remaining arguments are specific to each individual caller.

The flow of control for the task-related user exit interface is shown in Figure 13.

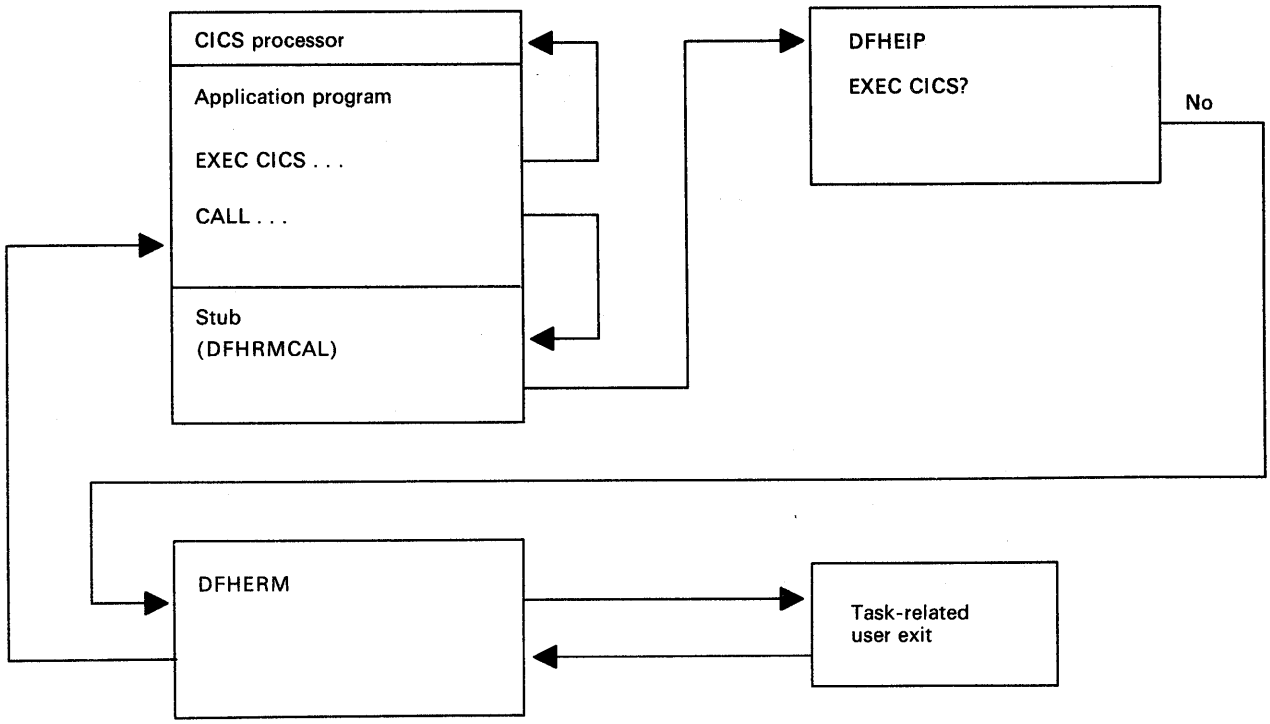


Figure 13. Task-Related User Exit Control Flow

Control Blocks

In the CSA, CSAUETBA points to the user exit table (UET). In the UET, UETHEPBC points to the first exit program block (EPB) and EPBCHAIN in each EPB points to the next EPB in the chain. Each EPB holds among other things:

- The address of the exit's entry point
- The address of the global work area
- The halfword length of the global work area
- The halfword length of the task-local work area.

For full details of the EPB, see the *CICS/MVS Data Areas* manual.

In the TCA, TCATIEBA points to the first transaction interface element (TIE), and TIECHNA in each TIE points to the next TIE in the chain. Each TIE holds among other things:

- The address of the exit's entry point
- The address of the global work area
- The address of the halfword length of the global work area
- The address of the halfword length of the task-local work area
- The address of the task-local work area.

For full details of the TIE, see the *CICS/MVS Data Areas* manual.

Processors

The term processor is used to refer to two different types of object:

1. For the EXEC interface, it refers to the function-dependent module(s) associated with the EXEC interface nucleus DFHEIP. These processors have names such as DFHETC, DFHEFC, DFHETS, DFHETD, etc., and each of these is invoked by DFHEIP. DFHERM is also a processor of this type.
2. In various contexts, including task-related user exits, it refers to a piece of code that is

link-edited with an application program and serves the dual function of:

Satisfying the CALL requirement for a target address – its entry resolves a V-type ADCON

Finding the entry point of DFHEIP.

Both of these types of processor are part of the path between an application call and the functional management module that supports the request. This path looks as follows:

```
Application call
 Application processor (type 2)
  DFHEIP
   EXEC interface processor (type 1)
    Functional management module
```

Examples of the interface are:

```
EXEC CICS SYNCPOINT  CICS's API
 DFHECI0    CICS's COBOL EIP router
  DFHEIP
   DFHESP   CICS's sync point router
    DFHSPP  CICS's sync point manager

EXEC DLI TERM ... DLI HLPI
 DFHECI0          CICS's EIP router
  DFHEIP
   DFHERM         CICS's non-CICS
                   router
    DLZEIP        DL/I's HLPI manager
                   (exit)
```

Interval Control Function

Interval Control Services Performed Independently of an Application

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

CICS Partition Exit Time

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

System Stall Detection and Correction

System stall detection and correction provides automatic detection of the system stall condition when the CICS 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

This mechanism automatically detects some cases when an application task is looping because of a failure in the program logic.

To be considered as a runaway task, an application program must have been in control for longer than a set time limit, and have made no calls to CICS that could allow selection of another task. The time limit is set by the user by means of the ICVR parameter in the system initialization table (SIT), or by the CEMT SET RUNAWAY command, and is in units of TCB time.

The action taken by the system is to abend the user task with the abend code AICA.

Interval Control Services Initiated by an Application

The following services are performed by interval control in response to a specific request from either an application program or another CICS function:

Time of Day

The EXEC CICS ASKTIME command retrieves the current time-of-day in either binary or packed decimal format.

Time-Dependent Task Synchronization

Time-dependent task synchronization provides the user with three optional services:

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

Automatic Time-Ordered Transaction Initiation

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 sending 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.

Interval Control Module (DFHICP)

Figure 14 on page 65 shows the relationships between the components of interval control.

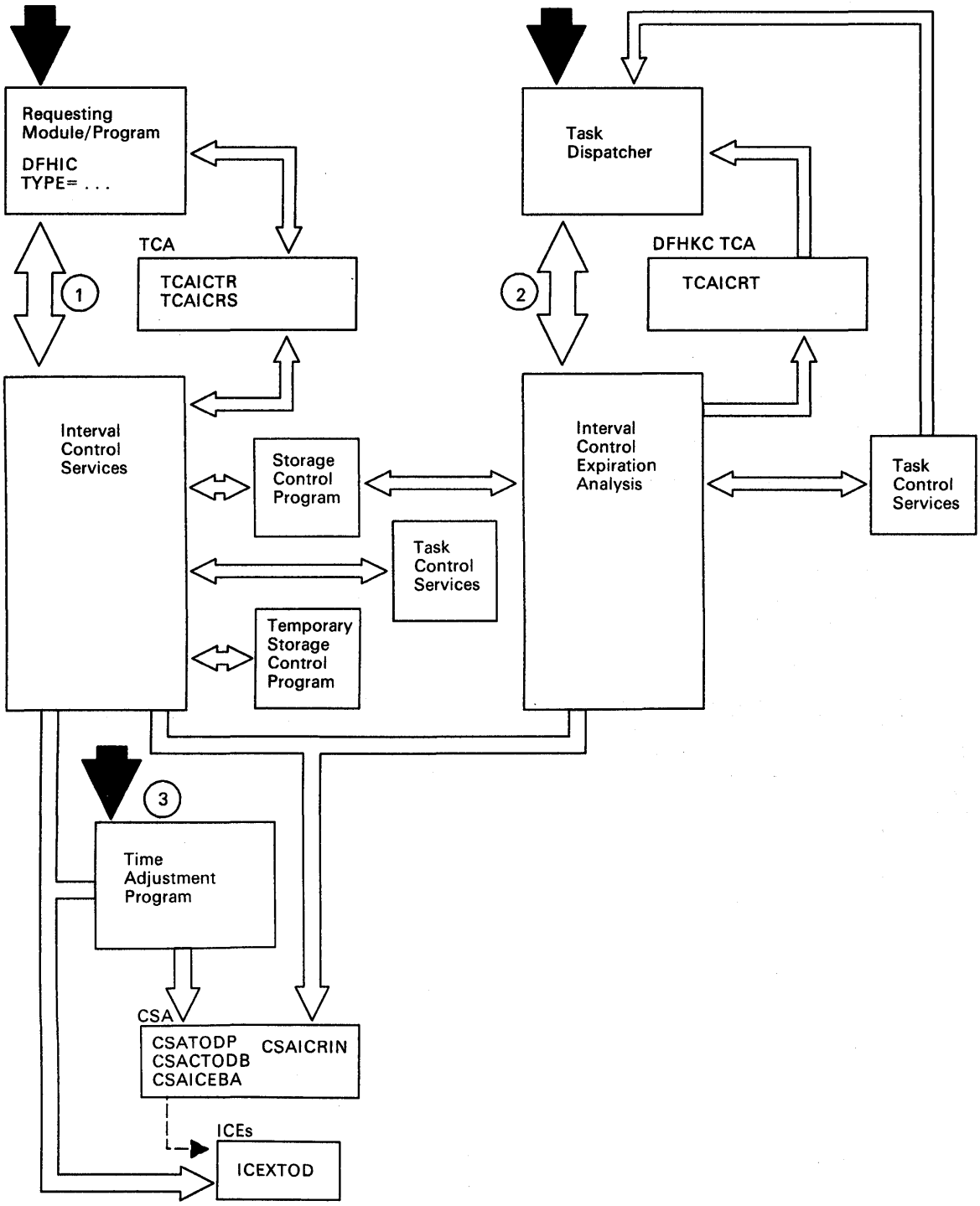


Figure 14. Interval Control Interface

Notes:

1. *Issuing a DFHIC macro instruction causes a type-of-request code to be set in TCAICTR, a field in the requesting task's TCA. Processing enters the interval control macro instruction service logic. The register contents are saved in TCAICRS and the requested service is performed or initiated. While performing the service, interval control interfaces with other CICS management functions (storage control, task control, and temporary storage control). Interval control builds interval control elements (ICEs) for time-dependent requests made by DFHIC macro instructions. The ICEs are chained off the CSA (CSAICEBA) in expiration time sequence (ICEXTOD). Control is returned to the requesting CICS module or user-written program when the requested service has been queued or performed. The DFHIC TYPE=RESET macro service (used to resynchronize the CICS time-of-day with the operating system) interfaces with the timer facilities of the operating system.*

2. *The task dispatcher activates task control's TCA for intermodule communication and branches to the expiration analysis logic of the interval control program. This logic examines the ICE chain entries, which are in expiration time sequence, until the first non-time-expired ICE is reached. Provided certain conditions do not exist, for example, any unfinished postinitialization programs, 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, interval control interfaces with storage control and uses task control services as well. Since the normal processing flow in the task control program is from the requested service logic, the expiration analysis logic is reentered by again activating task control's TCA as described above. The time remaining until the next ICE expires is returned in TCAICRT.*

   One of the functions of the task dispatcher is to continually refresh the current time-of-day values retained in the CSA (CSATODP and CSACTODB). During this process, resetting the clock at midnight can be detected.

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

Terminal Management Function

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)
- Graphics access method (GAM)
- Telecommunications access method (TCAM)
- Virtual telecommunication access method (VTAM), with the network control program (NCP).

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, or dynamically as needed. 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 is generated to satisfy the user's 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 task management ATTACH macro instruction is issued by terminal management. When data is to be transmitted to a terminal, processing programs execute a CICS 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 system to another or to another subsystem.

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, those lines are no longer dedicated to the CICS region. Thus 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 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 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.

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.

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. If the terminal has not been defined, VTAM passes information to CICS for the autoinstall of the terminal.

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 possible read and write operations have been initiated, terminal management processing is complete and control is returned to task management to allow dispatching of other tasks.

Transmission Facilities – TCAM

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

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 and additional user coding. CICS cannot arbitrarily take all action with regard to these errors. User application logic is sometimes 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, 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 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 can 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.

Terminal Management Modules (DFHTCP, DFHZCP)

Terminal management consists of two CICS modules, DFHZCP (which, for usability, is split into submodules) 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 system management functions (task management, storage management) CICS application services (basic mapping support and data interchange program), system reliability functions (abnormal condition handling), and with operating system access methods (BTAM, SAM, GAM, TCAM, or VTAM). 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 or operating system functions are issued by either DFHZCP (VTAM support) or DFHTCP, depending upon the terminal being serviced.

DFHZCP and DFHTCP 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 communication system.

Figure 15 on page 71 shows the relationships between the components of terminal management.

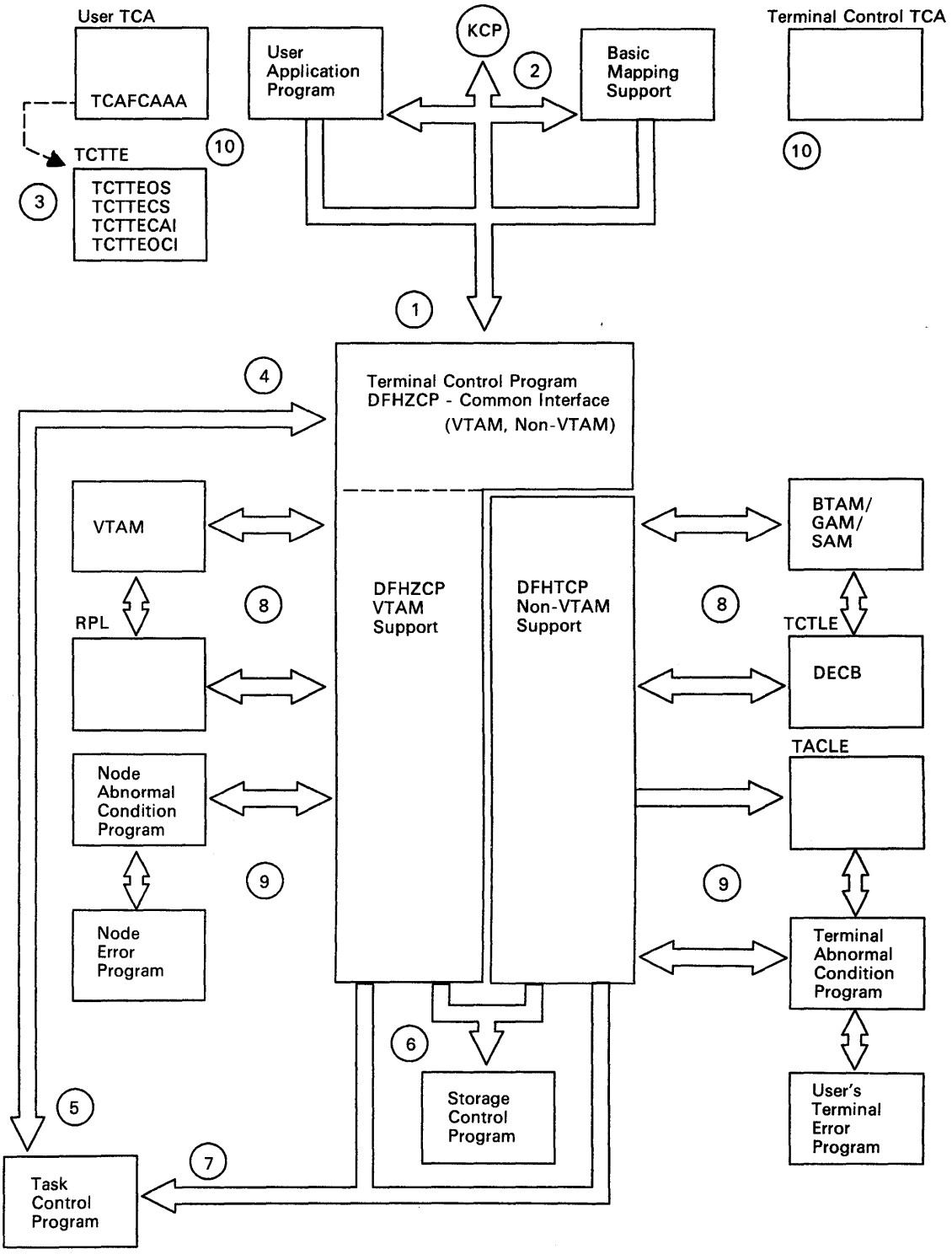


Figure 15. Terminal Management Interfaces

Common Interface

Notes:

1. *When a terminal management (DFHTC) macro instruction is issued by an application program or by the basic mapping support program (BMS), request bits are set in the user's TCA (transaction control area) and control is passed to the common interface (VTAM, non-VTAM) routines of DFHZCP.*

2. *If the DFHTC macro instruction includes a WAIT request and the IMMED option is not in effect, control is passed to task control to place the requesting program (task) in a suspended state. If a WAIT request is not included, control is returned to the requesting task.*

3. *The task's TCA contains either a field named TCAFCAAA (Facility control area associated address) a pointer to the terminal with which the task is associated, or a field named TCATPTA in which the address of the TCTTE to be used may be passed.*

4. *Task control dispatches terminal management through the common interface for one of the following reasons:*

   a. *The system address space exit time interval (specified in the ICV operand of the DFHSIT macro instruction by which the system initialization table (SIT) is generated) has elapsed.*

   b. *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).*

   c. *The specified terminal scan delay (ICVTSD in the SIT) has elapsed.*

   d. *None of the above has occurred, but one second has elapsed since the last time terminal management was dispatched with a pending request to be serviced.*

5. *Terminal management through its common interface, requests task control to perform a CICS WAIT when terminal management has processed through the terminal network and has no further work that it can do.*

Access Method Dependent Interface

6. *Terminal management communicates with storage management by means of 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

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

7. *Terminal management communicates with task management by means of the DFHKC macro instruction. The macro instruction can be issued by any one of the CICS control modules, depending upon the terminal being serviced. Terminal management may request task management to perform one of the following:*

   a. *Attach a task upon receipt of a transaction identification from a terminal.*

   b. *Respond to a DFHKC TYPE=AVAIL request (a task control macro instruction documented only for system programming) when a time-initiated task is indicated for a terminal and that facility is available.*

8. *Terminal control 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, for example, LOGON or LOSTERM. VTAM passes parameter lists and does not always use the RPL.

When authorized path VTAM has been generated, communication with VTAM also occurs in SRB mode (using DFHZHPRX); DFHZCP uses the RPL with an extension to communicate with its SRB mode code. When an SRB mode RPL request is complete, DFHZCP calls the relevant exit and post the ECB, as indicated by the RPL extension.

9. *Terminal management communicates with the CICS 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

   DFHZCA attaches the node abnormal condition program (DFHZNAC) when an error occurs. DFHZNAC does some preliminary error processing and then passes control to the user's node error program (DFHZNEP). Upon the completion of the user's error processing, control is returned to DFHZNAC.

10. *Terminal management executes either under the user's TCA or its own TCA as follows:*

    User's TCA

    a. *During the application program interface*

    b. *During the interface with basic mapping support*

    c. *While performing nonchained VTAM terminal requests.*

    Terminal Control's TCA

    a. *When task control dispatches terminal control*

    b. *When terminal control issues a request to task control (DFHKC)*

    c. *When terminal control issues a request to storage control (DFHSC)*

    d. *While performing non-VTAM terminal I/O or chained VTAM terminal I/O.*

Since many devices are supported by CICS terminal management, the number of modules required to provide this support is significant.

Figure 16 on page 74 through Figure 20 on page 79 give an overview of the interrelationships and functions within terminal management.

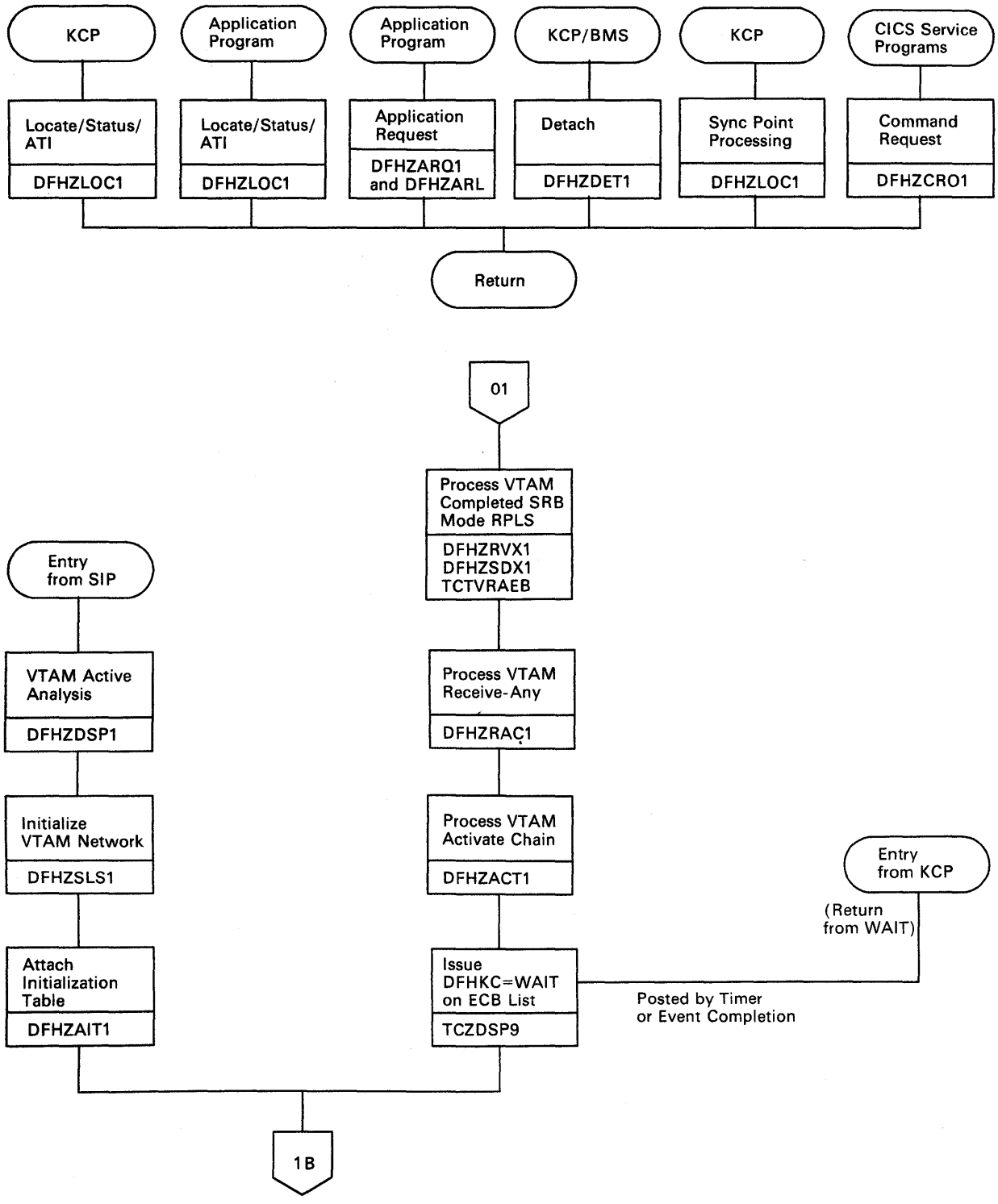


Figure 16 (Part 1 of 2). Terminal Management Common Control Routine

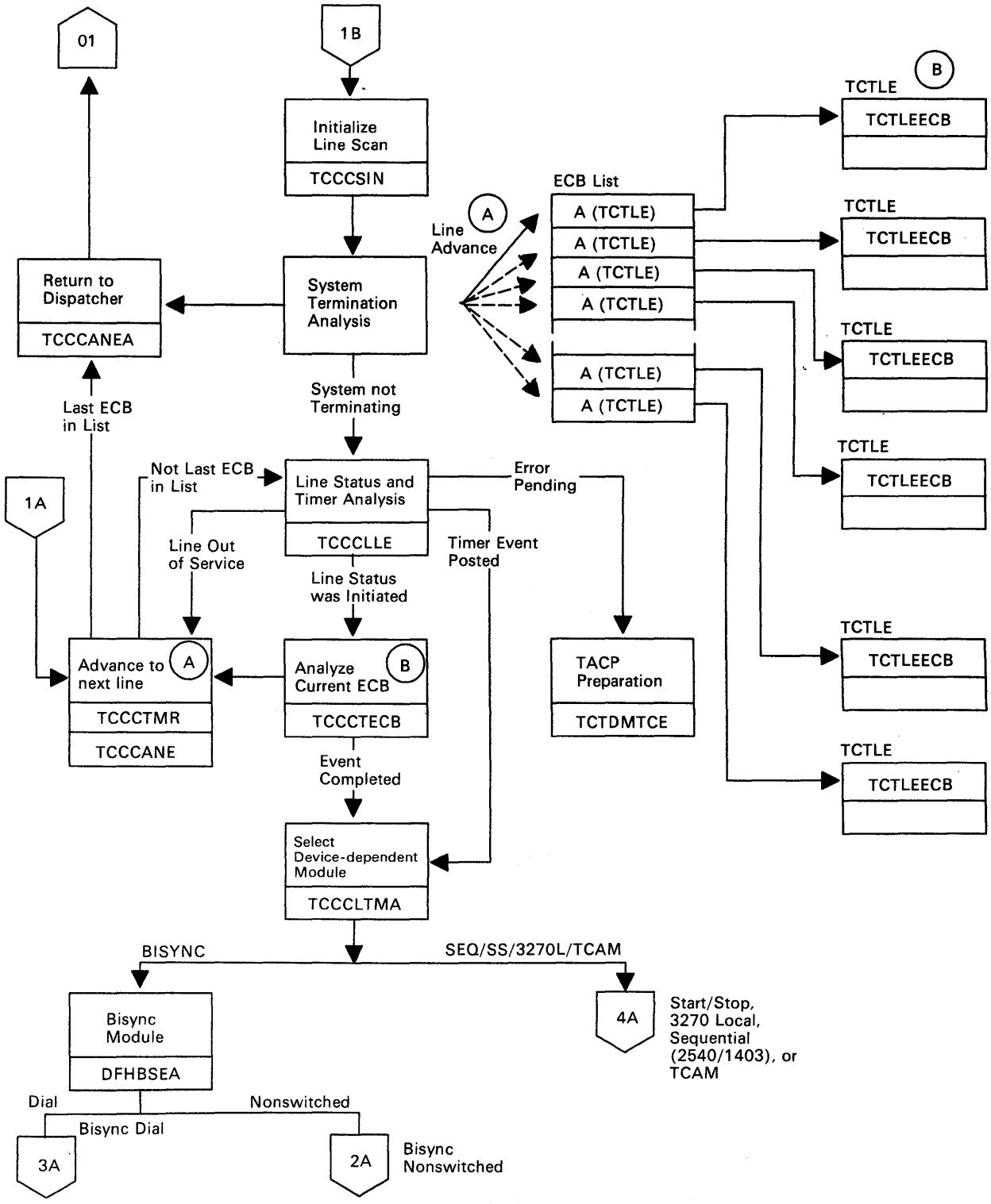


Figure 16 (Part 2 of 2). Terminal Management Common Control Routine

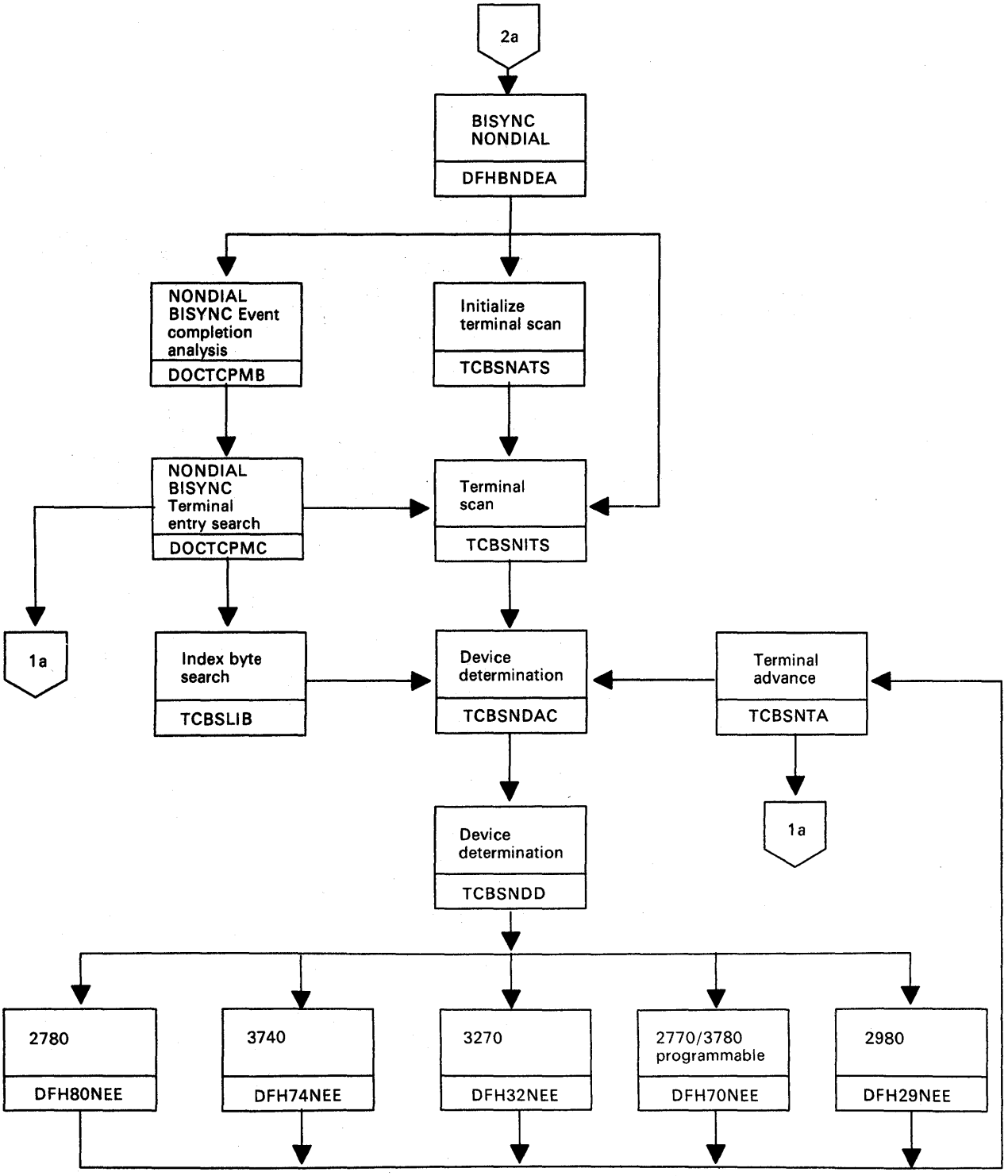


Figure 17. Terminal Management Bisync Nonswitched Routine

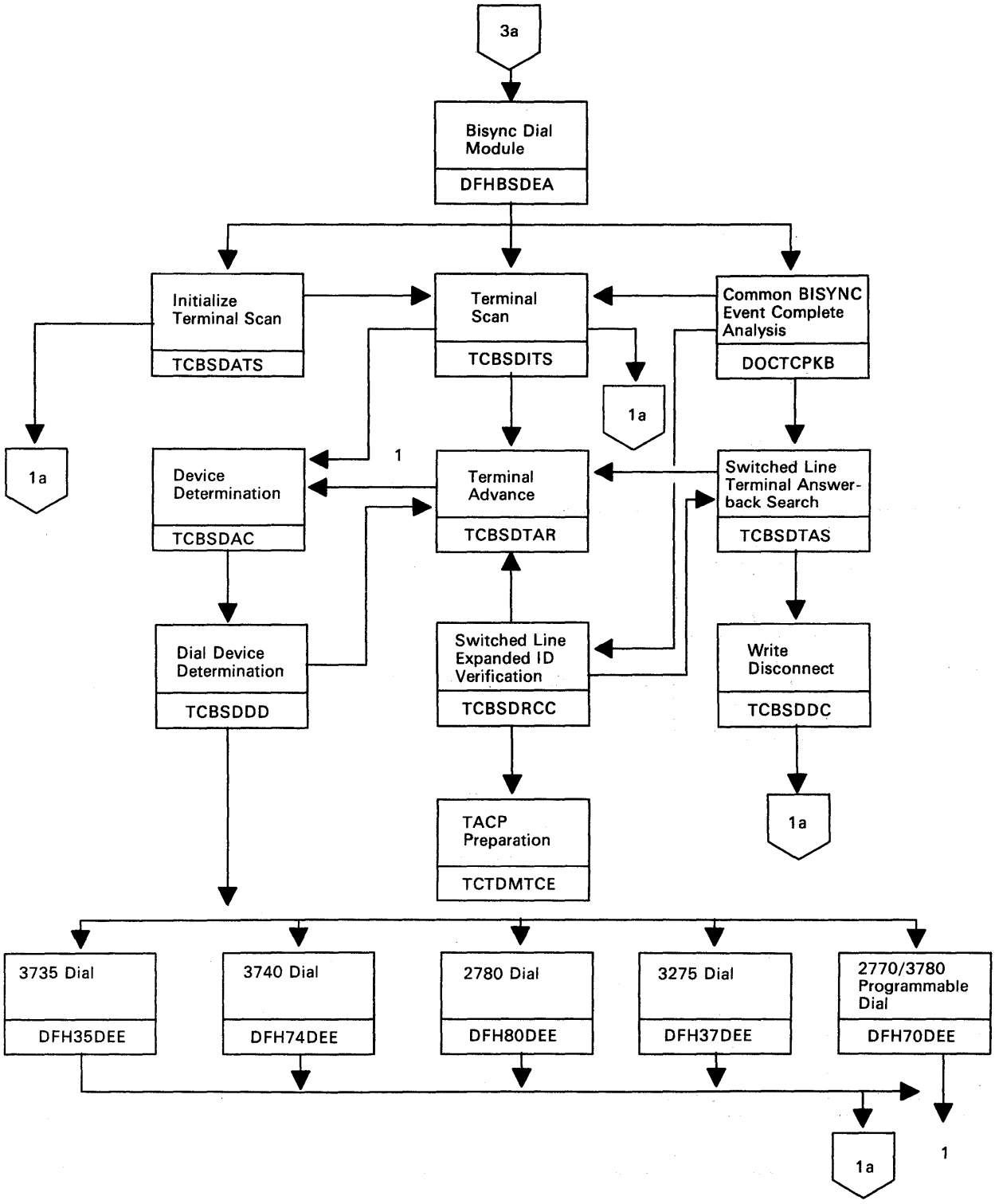


Figure 18. Terminal Management Bisync Dial Routine

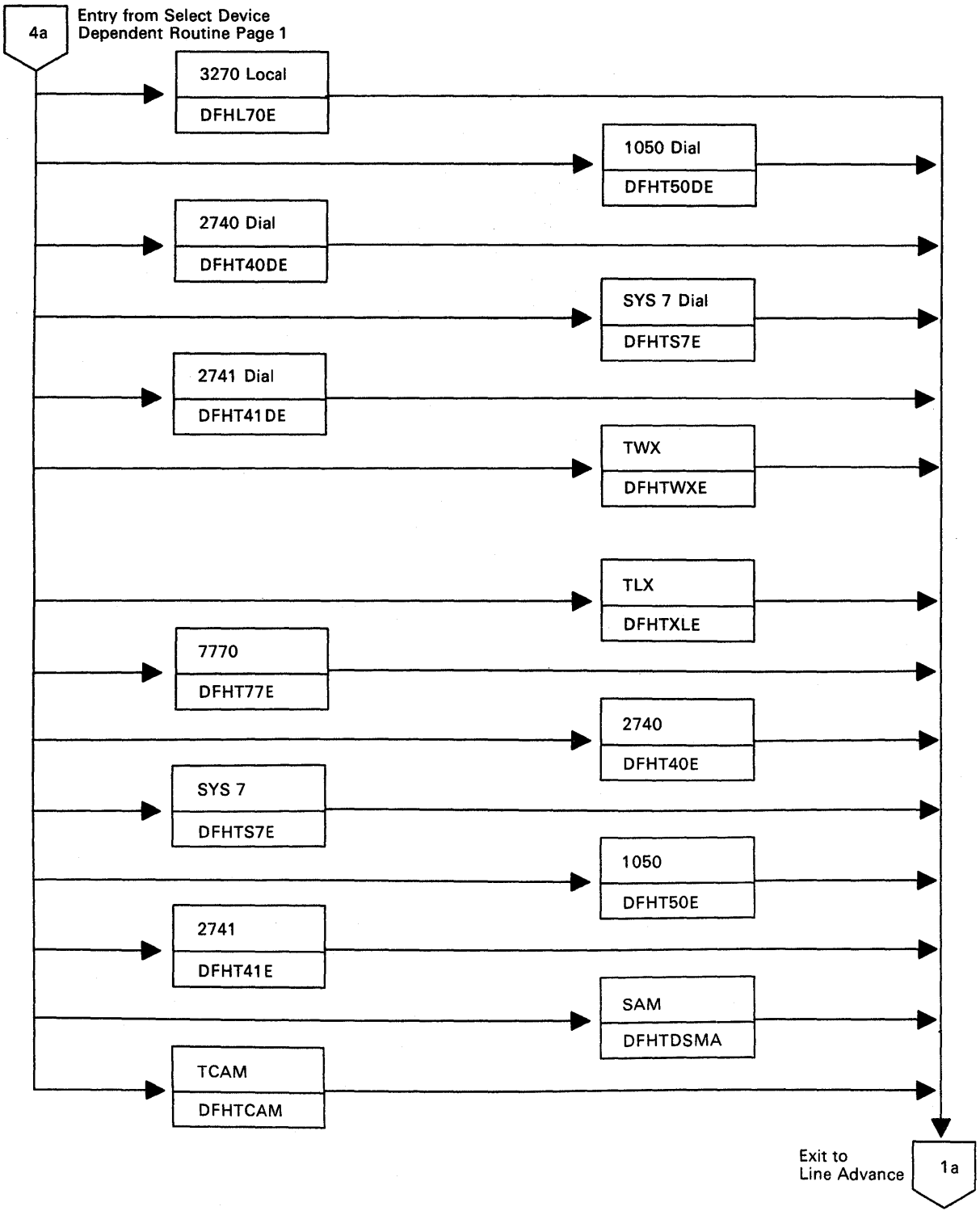
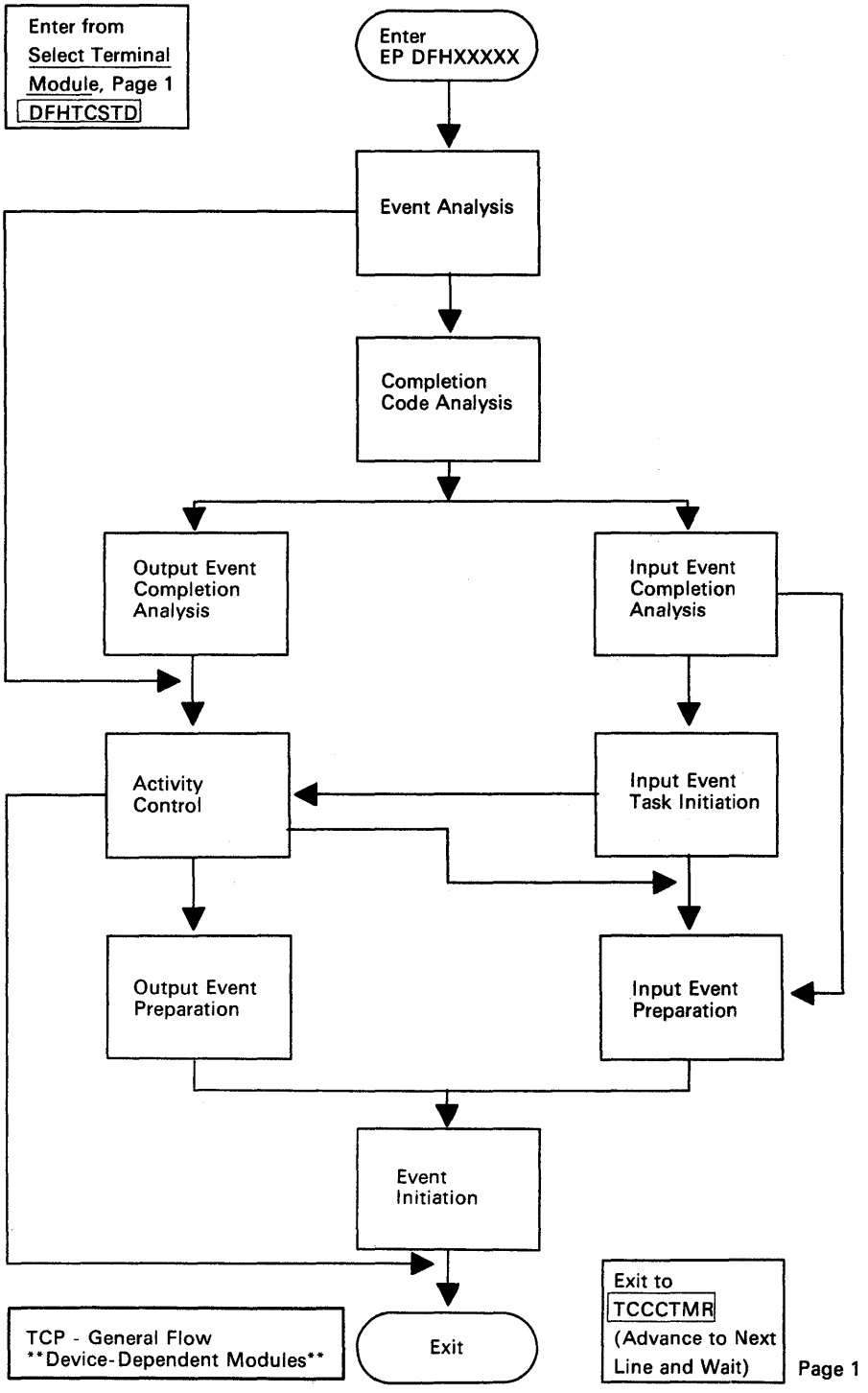


Figure 19. Terminal Management 3270L/SS Routines



DFHXXXXX

Device-Dependent Routine Entry Point Names	
Device Type	EP Name XXXXX
3735 Dial	35DEE
3740 Dial	74DEE
3740	74NEE
2780 Dial	80DEE
2770 Dial	70DEE
3270	32Nee
3275 Dial	37DEE
2770/3280	70NEE
2980	29Nee
3270 Local	L70E
1050 Dial	T50DE
2740 Dial	T40DE
SYS7 Dial	TS7E
2741 Dial	T41DE
TXL	TXLE
TWX	TWXG
7770	T77E
2740	T40E
SYS7	TS7E
1050	T50E
2741	T41E
SEQUENTIAL	TDMSA
TCAM	TCAMM

Figure 20. Terminal Management General Flow through Device-Dependent Modules

High Performance Option

When running CICS under MVS, the high-performance option (HPO) can be used. HPO uses VTAM as an authorized program so that the VTAM path length is reduced. This is achieved by dispatching SRBs to issue the reads and writes to the terminals. All the SRB code is executed in the module DFHZHPRX.

System Console Support

One or more OS/VS system consoles can be used as CICS terminals. Each console has a unique CONSLID specified in the TCT generation, causing a TCTTE and the associated CCE to be generated for each console. CICS communicates with the console using either the WTO macro or the WTOR macro.

A system console is defined to CICS by a TCTTE which has an associated control block, the Console Control element (CCE). The CCE holds the Event Control Block (ECB) for the console.

The interface between a system console and CICS is the Command Input Buffer (CIB), which is created in OS/VS protected storage for each MODIFY command. A CIB contains the data for a MODIFY command. CICS addresses the first CIB using the EXTRACT macro and the CIBs are chained together.

The OS/VS Communication ECB is in OS/VS protected storage and is posted complete for each MODIFY command and reset when there are no CIBs to be processed. The CICS system wait list holds pointers to the OS/VS Communication ECB and the ECB for each system console.

When CICS is initialized, an EXTRACT macro is executed to obtain the job name and point to the OS/VS Communication ECB and the first CIB; all these are stored in the TCT prefix.

DFHZCP contains two modules, DFHZCNA and DFHZCNR, which perform system console support.

DFHZCNA is used to:

- RESUME a task on completion of a terminal event for the task.
- ATTACH a task to satisfy a request for transaction initiation by a MODIFY command.
- ATTACH a task requested by automatic transaction initiation (ATI).
- DETACH a terminal from a task when the task has completed.
- SHUTDOWN console support when CICS is quiescing.

DFHZNCR is used to:

- Issue WTO macros for application program WRITE requests.
- Issue WTOR macros for application program CONVERSE or (WRITE,READ) requests.
- Issue WTOR macros with message DFH4200 for application program READ requests.

Console Support Management Modules

DFHZDSP calls DFHZCNA to scan the consoles for any activity.

DFHZCNA checks whether any task is suspended because it is waiting for a terminal event, for example, a READ, and, if the event is completed, resumes that task before starting any new task. This is done by scanning the CCE chain for ECBs which have been posted by OS/VS.

When a MODIFY command is executed, the communication ECB is posted complete and a CIB for the command is added to the end of the CIB chain. DFHZNCA processes the CIB chain in first-in, first-out order. For each CIB, DFHZNCA searches the CCE chain for the console ID.

The task is then attached if the "task pending" flag in the CCE is not set by a preceding CIB in the chain. In the course of scanning the CIB chain, DFHZCNA may find a MODIFY command that requires a task to be attached, but cannot do so immediately because there is already a task active, or there is an outstanding error condition to clear. It therefore sets the "task pending" flag in the CCE to remember the existence of the CIB. During the CIB chain scan, the condition preventing the task attach might clear, and a subsequent CIB might be selected for attach – the "task pending" flag prevents this, and ensures that CIBs are processed in order. All "task pending" flags are reset before each CIB chain scan.

If the task is to be attached, DFHZCNA obtains a TIOA and moves the data from the CIB to the TIOA. DFHZATT is then called to attach the task. If the attach fails, the TIOA is freed. A QEDIT macro frees the CIB if the attach is successful, and the scan continues.

When a transaction is automatically initiated and DFHKCP schedules the transaction for a terminal which is a console, a flag is set in the CCE by DFHZLOC. After DFHZCNA has completed scanning the CIB chain, it checks whether the console does not have a task already attached and there is not a CIB on the chain for the console; if both these conditions are satisfied, then the task is attached.

DFHZCNA issues a QEDIT macro to prevent any more MODIFY commands being accepted when CICS is shutting down. Any MODIFY commands on the CIB chain after shutdown has been started will be processed. When other access methods have been quiesced, and there are no tasks attached for a console, then console support will be shut down.

DFHZNCR sends terminal control requests from an application program to a system console by issuing WTO and WTOR macros.

If a console not defined to CICS is used to enter a MODIFY command, then DFHZCNA sets up an error code and links to DFHACP to issue the error message. This is done using the TCTTE for the error console, CERR. For a WRITE request, a WTO macro is executed with the MCSFLAG operand to specify the receiving console.

For a READ request, DFHZCNR executes a WTOR macro with a CICS-supplied message, DFH4200. This message requests the operator to reply, and the transaction waits for this reply. The MCSFLAG is used to specify the receiving console.

For a CONVERSE or (WRITE,READ) request, DFHZNCR executes a WTOR macro with the data specified for the WRITE. The transaction will then wait until the operator replies to this request. The MCSFLAG is used to specify the receiving console.

Defining Terminals to CICS

Terminal definitions are created as CSD records or DFHTCT macro instructions and then **installed** in (added to) the terminal control table (TCT) as TCT terminal entries (TCTTEs).

When a cold start is performed, CICS obtains its TCT entries from DFHTCT macros or from groups of resource definitions in the CSD file, which are named in the SIT GRPLIST operand. These are recorded in the CICS catalog.

When a warm start is performed, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored.

During CICS execution, TCT entries can be added as follows:

- By using the CEDA INSTALL command
- By the autoinstall process when an unknown terminal logs on
- By the transaction routing component when a TCT entry is shipped from a terminal-owning to an application-owning system.

DFHZCQ

Entries are installed in and deleted from the terminal control table by DFHZCQ. DFHZCQ is called by the following modules:

- DFHTCRP, during CICS initialization, to:
  - Restore terminal definitions at warm or emergency restart
  - Install RDO-eligible terminal definitions that are still defined by DFHTCT macros.
- DFHAMTP, for the CEDA transaction, to install TCT entries
- DFHZATD, the autoinstall program
- DFHZTSP, when a TCT entry is shipped for transaction routing
- DFHQRY, when the QUERY function is used to discover the actual characteristics of a device and complete the TCT entry for it
- DFHWKP, the warm keypoint program, to record information for RDO-eligible terminals in the CICS catalog.

It is important to note that DFHZCQ is heavily dependent on the module that calls it to supply the complete set of parameters to be used to create the TCTTE; DFHZCQ itself is not responsible for determining parameters for the TCTTE.

DFHBS... Builder Programs

DFHZCQ calls the builder programs, whose names all begin DFHBS. These **builders** are responsible for creating TCTTEs. The parameters given to DFHZCQ are passed on to the builders, which extract the parameters and set the relevant fields in the TCTTE.

To build, or delete, a control block for a particular device, a set of builders is called. The set of builders is specified by a tree structure of patterns, each pattern specifying one builder.

Builder Parameter Set (BPS): Each time a calling module invokes DFHZCQ, it supplies a builder parameter set (BPS). The BPS describes the device to be defined. The device-type is determined by matching attributes in the BPS with a table of definitions in DFHTRZYT.

A BPS consists of a fixed-length prefix, a bit map preceded by its own length, an area for fixed-length parameters preceded by its own length, and three variable-length parameters, BIND, USERID, and PASSWORD. Each variable-length parameter has a 1-byte length field.

DFHTBS

This component knows nothing, itself, about the TCTTE. It can be thought of as an interpreter that uses a pattern given to it by DFHZCQ to drive the whole install or delete process according to some simple rules. DFHZCQRT is the array of patterns, each of which specifies a list of builders that need to be called to create this particular type of TCTTE; each pattern is equivalent to a device-type. The array entry consists of:

- Information understood only by DFHZCQ
- The pattern that is interpreted by DFHTBS.

DFHTBSSP

This is primarily responsible for updating the catalog as a result of the request initiated by DFHZCQ and driven by DFHTBS.

Locks

Table management services (DFHTMP) is used to locate TCT entries. When DFHTMP gives the address of an entry, it notes the address of the calling task, and this has the effect of a shared lock.

When a TCT entry is deleted, it must not be in use by another task. This is achieved by issuing the DFHTM QUIESCE macro. Other tasks that issue DFHTM LOCATE for that entry will be suspended when they acquire a shared lock. These tasks are resumed when the original task issues a sync point to commit the deletion, or roll back.

Let us now look at each of the callers of DFHZCQ in turn.

System Initialization (DFHTCRP)

This component is responsible for reestablishing TCTTEs that were in existence in the previous CICS run. There are conceptually three stages of processing in DFHTCRP:

1. Initialize DFHZCQ; then exit if START=COLD
2. Reestablish TCTTEs recorded in the CICS catalog; then exit if START=WARM
3. Forward-recover TCTTEs recorded in the recovery log.

CEDA INSTALL (DFHAMTP)

When the CEDA INSTALL command is used to install a group of TERMINAL definitions, the flow of control is as follows:

- DFHAMP: processes CEDA commands
  - DFHAMPIL: processes the INSTALL command

    DFHAMTP: calls DFHTOR and then DFHZCQ.
- DFHTOR: receives as input a partial definition (TERMINAL, TYPETERM, CONNECTION, or SESSIONS), calling one of the DFHTOAxx modules, depending on the type of resource definition:
  - DFHTOAxx: adds a partial definition to a BPS. For a terminal device, a complete BPS is built from information from one TYPETERM and one TERMINAL definition; for an ISC or MRO link, a complete BPS is built from information from one CONNECTION and one SESSIONS definition.

    DFHTOBPS: builds the BPS, calling one of the DFHTRZxP modules, depending on the type of resource definition:

    DFHTRZxP: translates the parameter list into BPS format.
  - When DFHTOR has built a complete BPS, it returns it to DFHAMTP, ready to be passed to DFHZCQ.

Autoinstall

Autoinstall provides the ability to log onto an LU, known to VTAM but not previously defined to CICS, and to make a connection to a running CICS system. You should read the chapter on autoinstall in the *CICS/MVS Resource Definition (Online)* manual, for an introduction. Some of the facilities of resource definition online (RDO) are

necessary for autoinstall. There is further information in the *CICS/MVS Customization Guide* on implementing the **autoinstall user program**. The supplied program, which provides the basic function, is DFHZATDX; the reason why it is user-replaceable is that you may need to tailor the basic function to suit your system. In this chapter, we describe the flow of control when a terminal logs on and when it logs off. Later, we describe the flow of control when a terminal definition is shipped to another system.

Logon Flow

When an LU attempts to log on, VTAM drives the logon exit, DFHZLGX (in load module DFHZCY). DFHZLGX searches for the terminal in the TCT by comparing the NETNAME passed by VTAM to the NETNAME found in each NIB descriptor generated in the TCT. It is when a match is not found that CICS verifies that the LU is eligible for autoinstall. If it is eligible, an autoinstall work element (AWE) is built using an MVS GETMAIN for subpool 1.

The AWE is stored for later processing, when it is processed by DFHZACT attaching transaction CATD.

This transaction invokes DFHZATD, which retrieves suitable model definitions from the CICS catalog, and invokes the autoinstall user program (see above). This determines the TERMINAL name, selects one model from the list, and, optionally, sets the associated PRINTER and ALTPRINTER attributes. Then, control is passed back to DFHZATD. The rest of the information for creating the TCT terminal entry (TCTTE) is taken from the selected model TERMINAL definition and its associated TYPETERM.

Logon Exit (DFHZLGX) : Under the following circumstances, an LU is a candidate for autoinstall:

- If it is not already defined to CICS (using DFHTCT macros or RDO)
- If neither CICS nor VTAM is quiescing
- If the autoinstall user program (specified by the SIT AUTINST operand) exists
- If the VTAM RPL is present
- If it is not an LU 6.1 or LU 6.2 parallel session
- If it is an LU 6.2 single session terminal and ISC = YES in the SIT
- If the maximum number of concurrent logon requests (specified by the SIT AUTINST operand) has not been exceeded.

Processing consists of:

- Allocating an autoinstall work element (AWE)
- Copying the CINIT RU into the AWE
- Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

Activate Scan (DFHZACT)

- For every AWE on the AWE chain, dispatch the autoinstall program DFHZATD, passing it the address of the AWE.

Autoinstall (DFHZATD)

- Validate the BIND image in the CINIT RU
- Build a list of installable types from the autoinstall model TERMINAL definitions. (These must be have been installed, with appropriate TYPETERM definitions, either at system initialization or using CEDA INSTALL. When installed, these definitions are stored in the CICS catalog.) For an autoinstall of an LU to be successful, the following things *must match*:
  - CINIT BIND image, taken from the VTAM LOGMODE entry specified for the LU in the VTAMLST
  - Autoinstall model BIND image, built according to the specifications in the TYPETERM and TERMINAL definitions

  (Both versions of the BIND image should accurately define the characteristics of the device.)

  If the list is empty (no matching models are found), the request is rejected with an 0801 sense code returned to the LU and message DFH2411 is written to CSMT.

- Invoke the autoinstall user program
- Issue TC_INSTALL to create the TCTTE
- If INSTALL was successful, COMMIT the TCTTE and queue it for LOGON processing
- Free the AWE.

Most autoinstall problems can be grouped into two categories:

- CICS rejects the logon request, with message DFH2411
- The device rejects the actual BIND parameters.

To understand the reason why CICS rejects a logon, you need a VTAM buffer trace that shows the CINIT RU passed to the logon exit, and an IDCAMS printout of the CICS catalog.

The length of each BIND Image is found in the halfword preceding the image. A compare will be made for the length of the smaller value not to exceed 25 bytes (`X'19'`) bytes. The compare is accomplished by an XC (exclusive or) of the two BIND images into a work area. The result is ANDed with a mask that defines the required settings.

Additional bits are reset if the LU type, found in byte 14 of the BIND image, is 1, 2, 3, or 4. The final result in the work area must be 256 bytes of `X'00'`; any other value will cause CICS to reject the logon and return a sense code of 0801.

The second type of problem arises when the two BIND Images match, but the BIND is rejected by the actual device (sense code 0821). The *IBM 3274 Control Unit Description and Programmer's Guide*, GA23-0061, should be consulted for valid BIND parameters.

Disconnection Flow (LU Initiated)

VTAM Asynchronous Command Exit (DFHZASX): Driven by Request_Shutdown command

- Set on the `CLSDST` and `SHUTDOWN_IN_PROGRESS` flags in the TCTTE
- Put the TCTTE on the **activate chain** for DFHZACT to dispatch

Node Abnormal Condition Program (DFHZNAC)

- If task is still active, issue message DFH2470

Close_Destination (DFHZCLS)

- Set on the `PENDING_DELETE` flag in the TCTTE to prevent VTAM exits scheduling requests for the device.
- Issue UNBIND (CLSDST POST = RESP) for the device

Close Destination Exit (DFHZCLX)

- If CLSDST successful (that is, there is a positive response from UNBIND)
  - Set on the `SESSION_CLOSED` flag in the TCTTE
  - Flag the TCTTE for deletion
  - Enqueue the TCTTE to DFHZNAC.

DFHZNAC, Again

- Set on the `DELETE_REQUIRED` flag in the TCTTE
- Put the TCTTE on the activate chain for DFHZACT to dispatch

Autoinstall Program (DFHZATD): On the delete request, DFHZATD is invoked with the address of the TCTTE as input. Its function is to perform clean-up operations, the principal one being to ask DFHZCQ to delete the TCTTE.

Up to 3 attempts are made to delete the TCTTE. This is because the reason for the failure may be

the existence of a transient condition, such as the TCTTE being on the ZNAC queue to output a message to CSMT. Should the initial delete attempt fail, it is re-attempted after one second; if this fails, another attempt is made after a further 5 seconds. If the third attempt fails, it is assumed that the failure is a hard failure, such as an outstanding AID, which will not disappear until the device is reconnected; in this case, message DFH5943I is issued, SYNCPOINT takes place, and the TCTTE delete status is reset to make the TCTTE reusable.

If the deletion is successful, the delete is committed, the autoinstall user program is invoked to permit any specific clean-up operations to take place, and message DFH5966I is issued.

Shipping a TCTTE for Transaction-Routing (DFHZTSP)

For transaction routing, a terminal can be defined by an entry in the terminal-owning region (TOR) with the SHIPPABLE attribute; this is shipped to any application-owning region (AOR) when the terminal invokes a transaction owned by (and defined to) that region. (The entry in the TOR could have been installed using CEDA INSTALL, the GRPLIST at system initialization, or autoinstall.)

The first time a transaction is invoked

- In the AOR:
  - Look for an existing skeleton TCTTE (REMOTENAME)
  - Issue ZC_INQUIRE to the TOR
- In the TOR
  - Send a BPS representing the TCTTE to the AOR
  - Set on the **SHIPPED** flag in the TCTTE
  - Set on the **SHIPPED** flag in the TCTSE for the AOR system
  - Rewrite each entry to the catalog
- In the AOR
  - Generate a local TERMINAL name
  - INSTALL the terminal
  - Set on the **SHIPPED** flag in the TCTTE
  - Set on the **SHIPPED** flag in the TCTSE for the TOR system
  - Rewrite each entry to the catalog

When an autoinstalled TCTTE in a TOR is deleted

- If the deleted entry is flagged as having been shipped, notify all remote systems which have received shipped definitions that this terminal is being deleted
- Determine from the TCT in the AOR whether a definition for this terminal has been shipped
- Send name in the TOR to the AOR
- Convert to local TERMINAL name in the AOR
- Call ZC_DELETE in the AOR.

QUERY function (DFHQRY)

The QUERY function (DFHQRY) is used to determine the characteristics of IBM 3270 Information Display System devices, and complete the information about a device in the TCTTE. DFHQRY sends a read partition query structured field to the device, and analyzes the response. The TCTTE fields mainly affected are those used by basic mapping support (BMS), such as extended attributes. If QUERY(ALL) or QUERY(COLD) is specified in the terminal definition, DFHQRY is executed before any other transaction is initiated at a terminal. If QUERY(ALL) is specified, this is done after each logon. If QUERY(COLD) is specified, it is only done following the first logon after a cold start.

File Management Function

Access to the user file is provided by file management, which consists of the file management modules, and the file control table (FCT). File management reads from, and writes to, user-defined data sets, gathers statistics, and acquires dynamic storage for I/O operations. File management uses control information defined by the user in the FCT. This table describes the physical characteristics of all the 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
- BDAM, basic direct access method.

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
- BDAM deblocking
- Enabling and disabling of files, making them accessible to applications
- Opening and closing of files for the access method
- Exclusive control of records during update operations
- LOCATE mode, read-only retrieval (VSAM only)
- Mass record insertion (VSAM only)
- Automatic journaling.

Deblocking Services for BDAM Data Sets

CICS provides deblocking of logical records on a direct-access (BDAM) 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.

Exclusive Control

The access methods provide protection to the user against the concurrent updating (including addition or deletion) of a data set record by two or more transactions. This protection is called exclusive control.

Several transactions can concurrently update records on the same VSAM 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. 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.

Exclusive control is only maintained while a record is being updated, and is released as soon as the operation is complete. This contrasts with CICS record locking support, which does not release an updated record until the end of the logical unit of work.

CICS provides an enqueue lock on an update request which is held to the end of a logical unit of work (LUW). The user specifies that a file should be recoverable, and that record updates should be locked in this way by the LOG=YES option in the FCT.

Sequential Retrieval

Another optional facility of CICS file management is the sequential retrieval of records from the data base. This facility 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 application, while in browse mode, can issue a random get for update requests to a different data set, 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 Control User Exits

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 has been 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 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. To allow journaled records to be associated with the appropriate data set (instead of with the CICS file name), a special record is journaled showing the current data-set allocation whenever it changed.

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 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.

Automatic journaling of input and output messages is not supported for transactions running on surrogate terminals. The automatic journaling is done on the system that owns the real terminals (assuming the remote transaction definitions specify the MSGJRNL option).

VSAM Interface

For a description of the VSAM interface, see the chapter "CICS-VSAM Interface," in the *CICS/MVS Problem Determination Guide.*

VSAM/BSAM Subtasking

The VSAM/BSAM subtask program, DFHVSP, performs VSAM or BSAM requests asynchronously with the processing of other CICS requests, providing overlapped processing in a dyadic, attached, or multiprocessor.

DFHFCP uses the DFHSQEM macro to create a queue of subtask queue elements (SQEs), each of which represents a request for VSAM or BSAM services. DFHFCP posts the VSAM /BSAM subtask ECB, VSCASECB, and issues a CICS WAIT on VSWACECB for the transaction.

DFHVSP removes all SQEs from the request queue and transfers them to the process queue. Each SQE is removed from the process queue in turn and the corresponding VSAM/BSAM request is issued. The SQE is then placed on the wait queue. When the process queue is empty, the wait queue is examined to determine if any requests (SQEs) have completed. If so, VSWACECB is posted, causing the main task to dispatch the transaction which continues processing. When there is no more work to do, DFHVSP waits for a pending request to complete, or another post from DFHFCP.

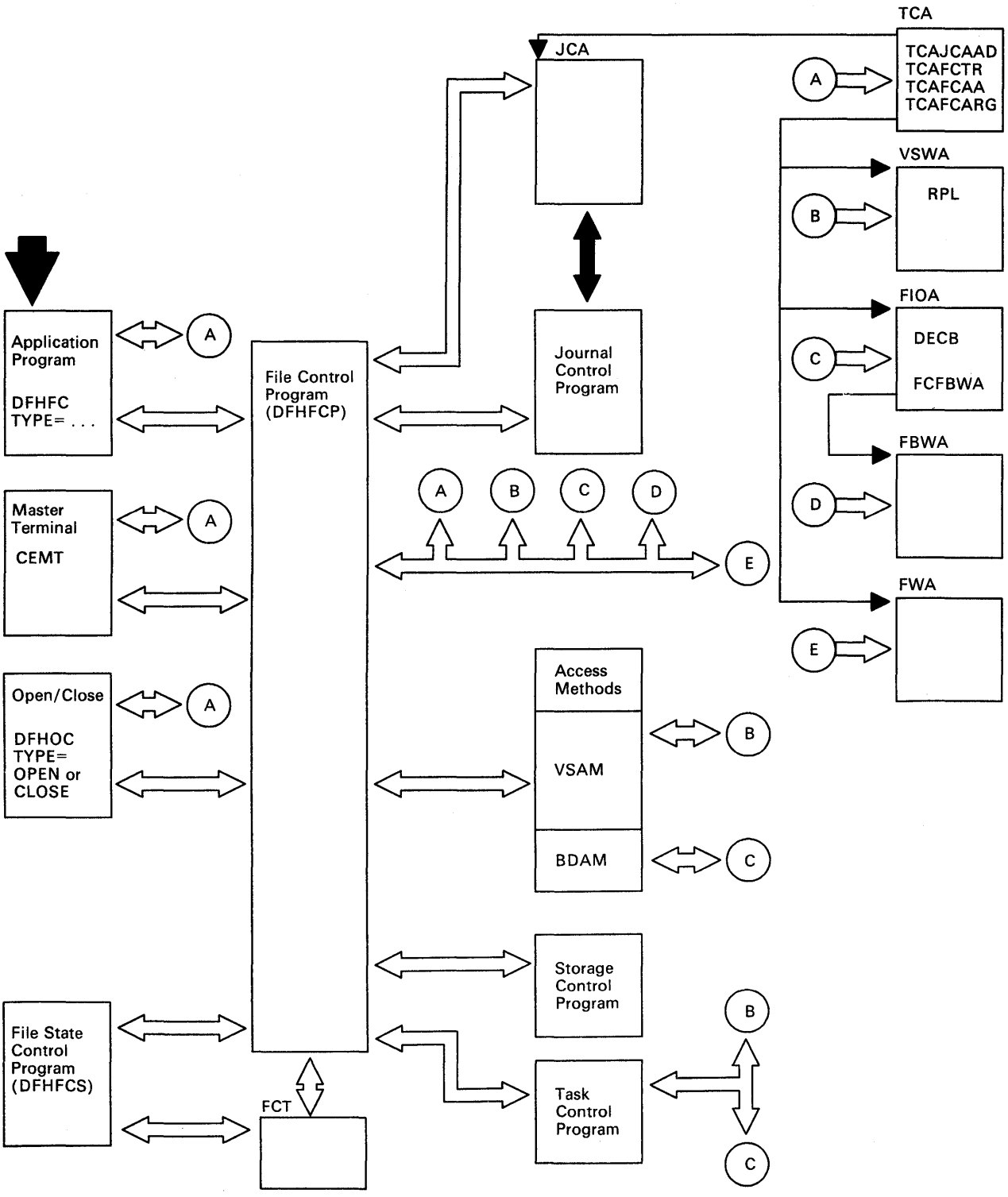


Figure 21. File Control Interface

File Management Modules (DFHFCP, DFHFCS)

The main file control programs are DFHFCP and DFHFCS. DFHFCP processes all accesses to VSAM and BDAM data sets. DFHFCS processes all file state changes, such as OPEN, CLOSE, ENABLE, and DISABLE.

The file control programs communicate directly with other CICS functions, the standard access methods, and user-written application programs (see Figure 21 on page 90).

Note: The on-page connectors in Figure 21 on page 90 show the relationships of the logical elements of the programs.

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

- Request storage control to acquire any required storage areas.
- Communicate with the standard access methods to request that any required I/O operations be performed.
- Request task control to place the application program (requesting task) in a wait state until I/O operations are completed as required.
- Request journal control to perform any automatic journaling required.
- Request DFHFCS to perform a file state change. DFHFCP links to DFHFCS, using the DFHPC TYPE=LINK macro. The field TCAFCARG, in the TCA, addresses a parameter structure which holds the information for the request.
- Return the address of the appropriate control block to the requesting task in the field TCAFCAA of the task's TCA.

DL/I Data Base Support

DL/I Support

Initialization

During CICS system initialization, DFHSIH1 invokes the DL/I interface initialization module (DFHDLQ) which prepares the environment for IMS/VS. If XRF = NO in the SIT, or XRF = YES and START ¬ = STANDBY, DFHDLQ performs all the initialization for DL/I. If XRF = YES and START = STANDBY, phases 1A and 1B are executed. In phase 1A, DFHSIH1 calls DFHDLQ, which calls IMS/VS to load the code and control blocks. In phase 1B, DFHSIH1 calls DFHDLRP, which calls IMS/VS to sign on to data base recovery control (DBRC), identify itself to the IMS/VS resource lock manager (IRLM), and complete the DL/I control blocks.

Note: These processes are done after takeover is complete, and before DL/I backout is started.

DFHDLQ then invokes the IMS/VS batch initialization interface (DFSRRC00) which initializes the IMS/VS environment and returns control to DFHDLQ which continues initialization. IMS/VS initializes the control block SCD and the required number of PSTs; DFHDLQ obtains storage for the ISB pool, and initializes the matching CICS ISBs.

The required number of PST/ISB pairs is given either by the DLTHRED option in the CICS system initialization table or by the DLTHRED override parameter in the PARM field of the EXEC card for the CICS startup. (A CICS ISB serves to link an IMS/VS PST with the TCA of a particular CICS task.)

If data sharing is in the system, DFHSIJ1 calls DFHDLX, which handles DL/I requests, to complete the initialization for DL/I. DFHSIJ1 also attaches the tasks DFHDLG to be an exit for the IMS/VS resource lock manager (IRLM) to drive global commands, and DFHDLS to be a status condition task when a local IRLM system abends, or when communication with a remote IRLM system fails.

Servicing DL/I Calls

To use an IMS/VS DL/I data base, an application program issues EXEC DLI statements, or DL/I CALL statements or DFHFC TYPE = DL/I macros (for scheduling, data base access, and termination). EXEC DLI statements are translated to standard DL/I call statements by DFHEDP.

The module DFHDLR holds the subroutines used by IMS/VS when IMS/VS requires CICS services such as logging, dispatching, or storage management. IMS/VS accesses the addresses of these subroutines from the system contents directory (SCD), just as in a non-CICS environment. IMS/VS initialization primes these addresses with the CICS subroutine addresses when working in a CICS environment.

SCHEDULING CALLS: On behalf of the calling task, DFHDLI locates an ISB/PST pair (or waits on the ECB DLISBECD until an ISB/PST pair is available). DFHDLI then invokes the IMS/VS module DFSDBLM0 to perform the scheduling operation. If the user's PSB indicates that an **update** operation is being scheduled, DFHDLI builds a scheduling log record (for possible backout purposes).

DATA BASE ACCESS CALLS: DFHDLI sets up the IMS/VS environment (by accessing the PST through the ISB) and invokes the IMS/VS program request handler DFSPRH00 to service the call. (Because DFSPRH00 requires a standard DL/I argument list, DFHDLI translates any DFHFC TYPE = DLI macro invocation into a standard DL/I argument list.)

After it has handled the call, DFSPRH00 returns control to DFHDLI which updates statistics and then returns to the calling program.

TERMINATION CALLS: A CICS task relinquishes its use of IMS/VS by issuing a TERM call to DFHDLI. DFHDLI performs the following operations:

1. If called by an application program, a call is made to DFHSPP followed by a return to the application program (DFHSPP calls DFHDLI recursively to perform the remainder of the operations).

2. The IMS/VS program DFSFXC50 is called with the phase I sync point option to purge the data base buffers.

3. If the PSB has been scheduled for update, a TERM (X'07') log record is written.

4. A DWE is created so that the TERM processing can be completed after DFHSPP has written an end-of-sync-point record on the system log.

5. A return is made to DFHSPP.

On return to DFHSPP, the DWE is processed. DFHDLI makes a phase II sync point call to the IMS/VS program DFSFXC50 to release all locks, and an RELPSB call to the same program to free the data base and the PSB.

Calls to Remote Data Base

If a call is for a remote data base, DFHDLI recognizes this fact and invokes DFHIS TYPE=CONVERSE to ship the call for remote (rather than local) processing.

The CICS-DL/I interface program (DFHDLI) communicates directly with user-written application programs, DL/I and other CICS functions. The DL/I interface accepts requests for DL/I processing from application programs as well as CICS service modules.

Data Sharing

Data sharing is the facility for a CICS system to share the same DL/I data base with other systems such as CICS, IMS/VS DB, or IMS/VS DC. The main program used for data sharing is DFHDLX. How data sharing works is described in the *CICS/MVS Problem Determination Guide.*

Dynamic Allocation and Deallocation

If using the dynamic allocation and deallocation facility of IMS/VS, allocation occurs automatically during the first schedule after system initialization, or when a CEMT SET DATASET(name) OPEN command is executed. Deallocation occurs automatically at system termination, or when a CEMT SET DATASET(name) CLOSED|RECOVERDB command is executed.

DL/I Interface Program

Figure 22 on page 94 shows the relationships between the components of the CICS-DL/I interface.

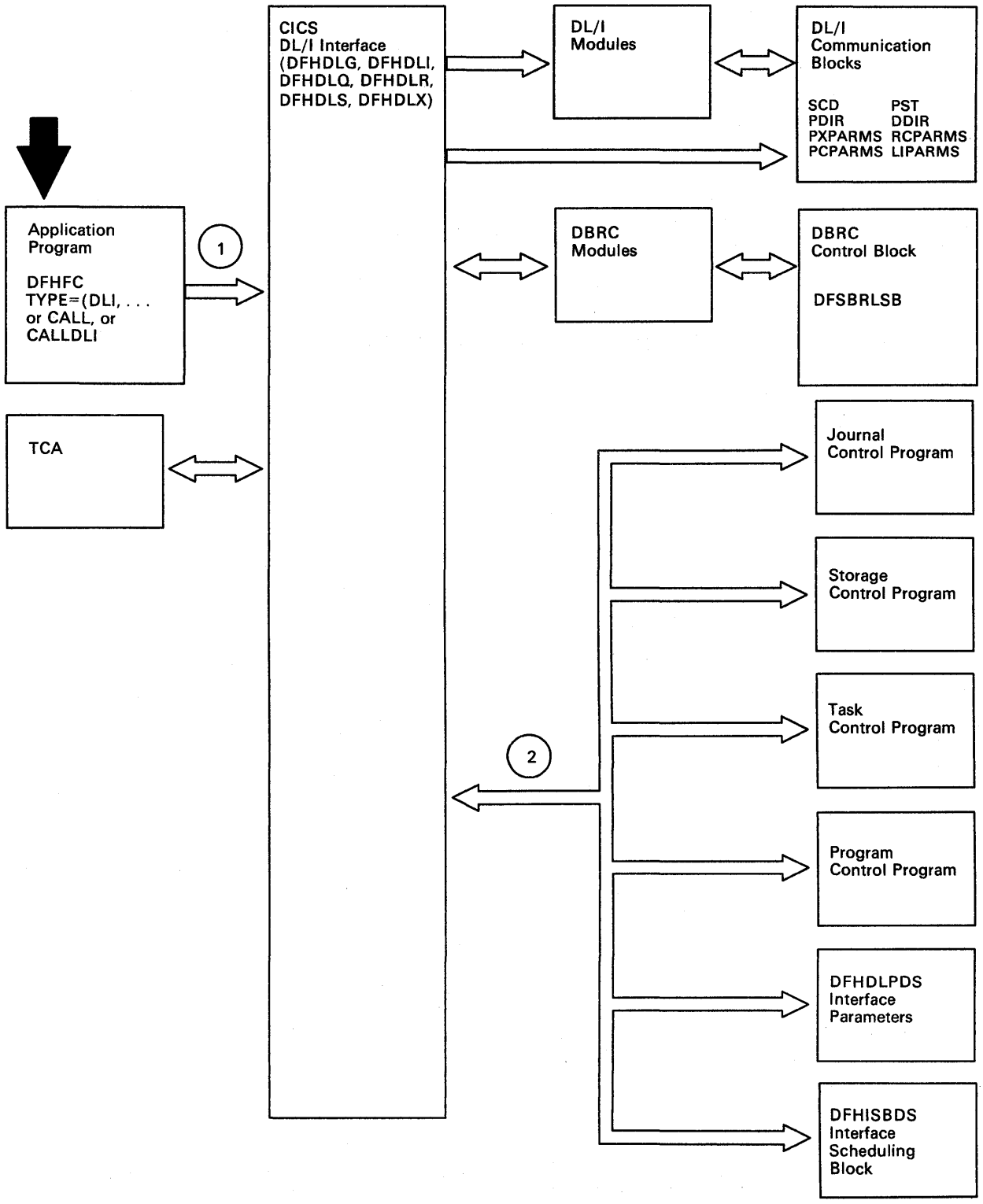


Figure 22. CICS-DL/I Interface

Notes:

1. *When DL/I functions are requested by an application program or a CICS management module through execution of a file control macro instruction (DFHFC TYPE=DLI, xxx), fields within the common communications area of the TCA are filled with appropriate entries to communicate with the DL/I interface. The functions can also be requested by a CALL or a CALLDLI macro instruction, in which case DFHDLI fills in the required fields in the TCA. EXEC DLI statements are translated to standard DL/I call statements by DFHEDP. If the request is for a local data base, DFHDLI communicates the requested function to IMS/VS service modules. If the request is for a remote data base, DFHDLI invokes the CICS function request shipping CONVERSE service to transmit the request to the appropriate system. In addition to processing DL/I input/output requests, DFHDLI on request, schedules and terminate program specification blocks (PSB).*

2. *In servicing the requests mentioned above, DFHDLI communicates these requests to DL/I modules and also uses various CICS modules. DFHDLI issues requests to journal control to perform logging functions, to storage control to obtain required storage, to task control to perform the WAIT function, and to program control in the event of abnormal termination. DFHDLI also uses the DL/I interface parameters and the DL/I interface scheduling block (ISB) for storing and retrieving control information. If the request is for a remote data base, then a Remote Scheduling Block (RSB) is used instead of the DL/I interface scheduling block.*

   DFHDLI calls DFHDLX for certain requests, including DBRC requests.

Transient Data Management Function

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 data set, for eventual input to one or more CICS transactions. Intrapartition destinations are accessible only by CICS transactions within the CICS 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. The time at which space becomes reusable is determined by the **reuse** bit in the destination control table (DCT) entry for the queue. If transient data management cannot free the space automatically, the space taken up by a queue continues to grow until the queue is purged by an application program.

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.

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 during normal execution. Two types of recovery are possible: **physical** and **logical**.

Physical Recovery of Intrapartition Transient Data Queues: The DCT entry is restored to the state it had when the system abnormally terminated. Nothing is backed out for in-flight transactions. The input pointers in the DCT entry are reset from the system log where each GET and PURGE was recorded. (To ensure that an input pointer is established on the log, even if no GETs or

PURGEs are subsequently recorded, the first PUT to an empty queue is always logged.) The transient data recovery program then reads the queue forward, and sets the output pointer to the end of the queue as discovered from the data set.

Logical Recovery of Intrapartition Transient Data Queues: The DCT entry is restored to the state it had at the last sync point. In-flight transactions are backed out. The DCT entry is logged at each sync point to provide a record of the DCT entry for this purpose if the logical unit of work has made any intrapartition requests.

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 for subsequent input to CICS or for offline processing. Sequentially organized data created by other than CICS programs can be entered into CICS 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 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 command in the application program is executed to retrieve the queued data. All data in the queue is retrieved sequentially for the application program.

Transient Data Services

The following services are performed by transient data management in response to CICS 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.

Transient Data Management Module (DFHTDP)

Intrapartition Destinations

Figure 23 on page 98 shows transient data control program interfaces for intrapartition destinations.

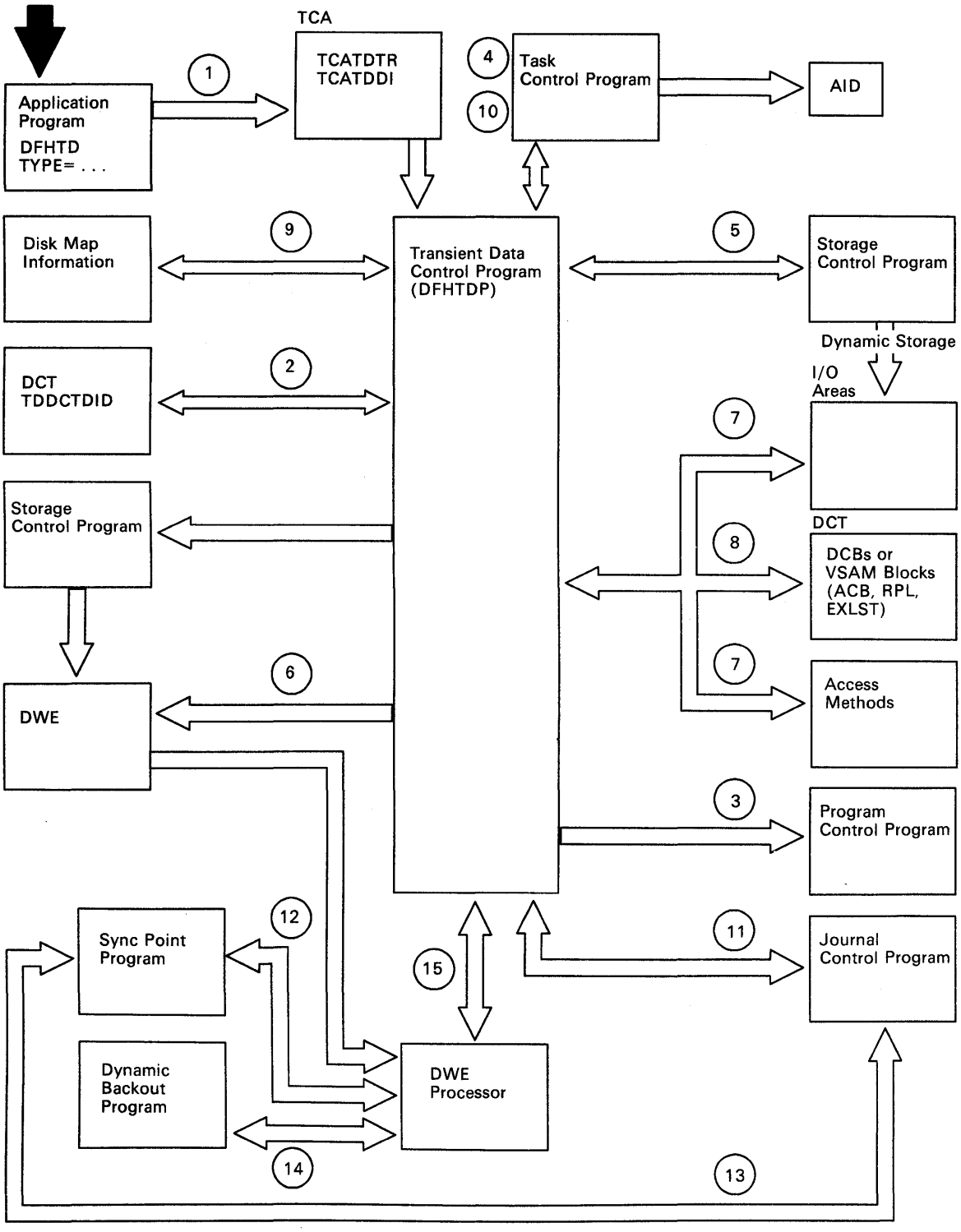


Figure 23. Transient Data Control Program Interfaces for Intrapartition

Notes:

1. *An application program request for transient data services causes a request code to be set in TCATDTR, signifying GET, PUT, PURGE, LOCATE. The destination identification is placed in TCATDDI. Control is passed to the transient data control program.*

2. *The field TDDCTDID is used to search the DCT for the destination requested.*

3. *If an invalid request for transient data services is received, control is passed to program control to terminate the task issuing the request.*

4. *Task control is used to:*

   a. *Ensure that only one GET, PUT, or PURGE request from a given queue can be processed at a time.*

   b. *Acquire, with logical recovery, ownership of an end of a queue until the end of the LUW.*

   c. *Wait for I/O completion.*

5. *Transient data control uses storage control to obtain storage for an intrapartition input area.*

6. *If logical recovery is supported by DFHTDP and requested for the destination, storage control or RPL creates a deferred work element (DWE) for each logically recoverable destination when it is accessed.*

7. *Transient data uses the VSAM/BSAM subtask program, DFHVSP, for input/output. (see "VSAM/BSAM Subtasking" on page 89 for more details).*

8. *The RPL status bytes are tested for I/O errors.*

9. *Transient data control obtains intrapartition disk map information from the main storage area addressed through the destination control table (DCT).*

10. *If an automatically initiated task is to be associated with a terminal, task control is used to schedule an AID. If an automatically initiated task is not to be associated with a terminal, task control is used to attach the task.*

11. *For physically recoverable destinations, the journal control program is used to log the DCT entry.*

12. *At the normal end of a logical unit of work (LUW), defined by either an application program DFHSP TYPE= USER request or by task control at task termination, the CICS sync point program gives control to the DWE processor of the transient data control program. This DWE processor performs the logical update of the DCT entry and/or the disk map. A separate DWE is used for GET, PUT, and PURGE for each logically recoverable destination accessed by the task. The CICS sync point program or the dynamic backout program (if dynamic backout required) presents each DWE separately to the DFHTDP DWE processor.*

13. *For logically recoverable (DWE processor) destinations, the journal control program is used to log the DCT entry.*

14. *If the task abends, DFHDBP invokes the DWE processor to perform transient data backout of the DCT and disk map when the dynamic backout program encounters a transient data DWE.*

15. *The DWE processor (part of DFHTDP) uses common routines in DFHTDP.*

Extrapartition Destinations

Figure 24 shows transient data control program interfaces for extrapartition destinations.

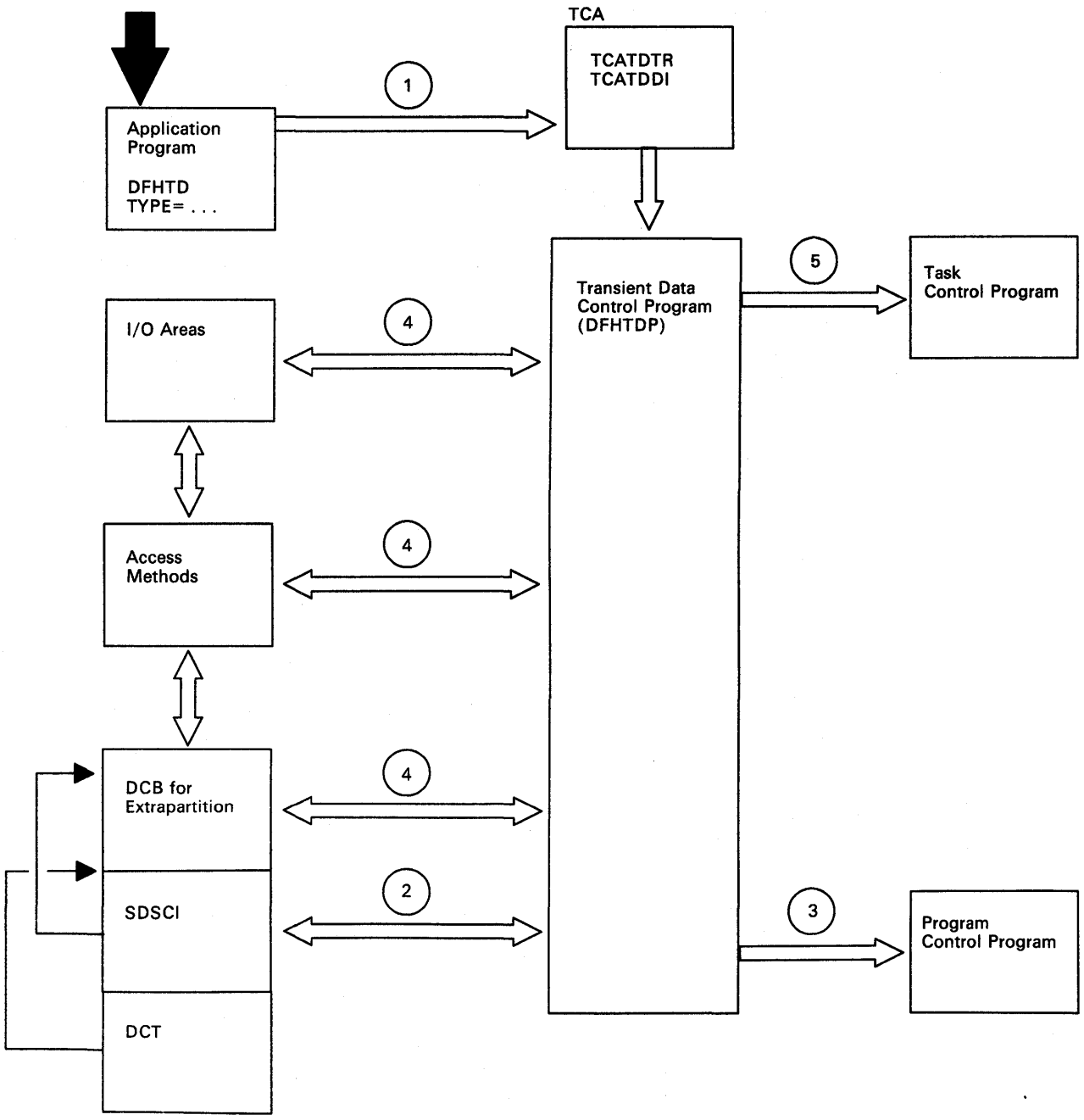


Figure 24. Transient Data Control Program Interfaces for Extrapartition

Notes:

1. *An application program request for transient data services causes a request code to be set in TCATDTR, signifying GET, PUT, FEOV, or LOCATE. The destination identification is placed in TCATDDI. Control is passed to the transient data control program.*
2. *The DCT is searched by field TDDCTDID for the destination requested.*
3. *If an invalid request for transient data services is received, control is passed to program control to terminate the task issuing the request.*
4. *The request is passed to the appropriate QSAM routine for processing.*
5. *The task control program is used to enter a dispatchable WAIT, before returning to the application program. For a GET request, the dispatchable WAIT is entered before control is passed to the appropriate QSAM routine.*

Temporary Storage Management

Temporary storage management provides the services necessary for an application program to store data temporarily in main storage or on a direct-access device. For MVS/XA, the main storage used is from above the 16 megabyte line. 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 main storage.

If the release of data is requested, temporary storage management frees the main storage or direct-access storage space that was used for the data.

Temporary storage management provides the basic services for the CICS terminal paging function.

Temporary Storage Management Services

The following services are performed by temporary storage management in response to CICS commands:

Temporary Storage Put
Allocates main 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 main 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.

Large Temporary Storage Records

If the length of the data plus the standard length field is greater than the length of a control interval of direct-access storage, the data is split into sections of length less than that of a control interval. The sections are written in order, followed by a control record. On reading the data, the control record is read first followed by the sections. For these operations, DFHTSP calls itself, and consequently any active user exits will be executed each time.

Temporary Storage Management Module (DFHTSP)

Figure 25 shows the relationships between the components of temporary storage control.

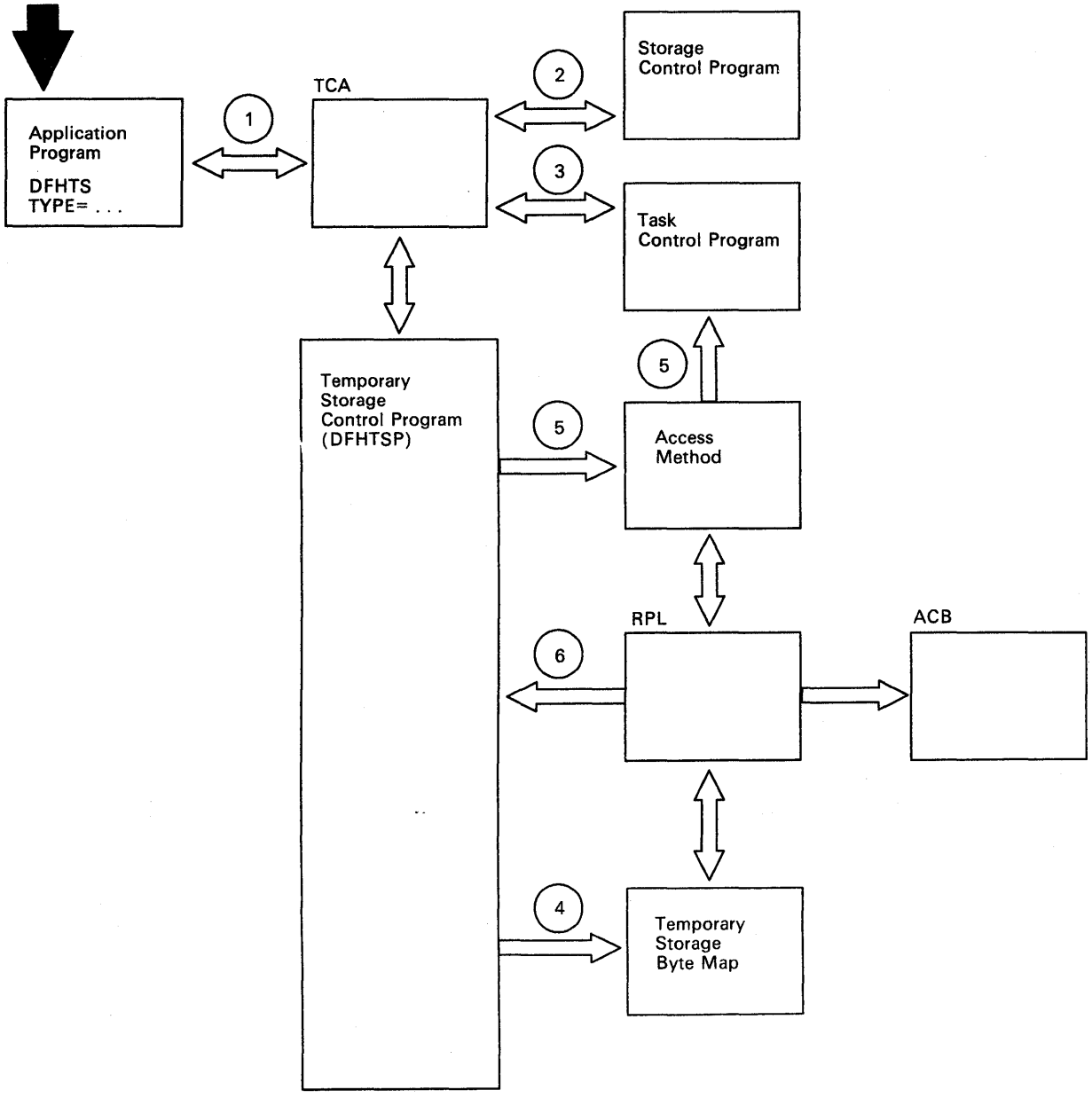


Figure 25. Temporary Storage Control Interfaces

Notes:

1. *An application program requests temporary storage services by means of a command.*
2. *Temporary storage control communicates with storage control 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.*
3. *Temporary storage control communicates with task control to perform a CICS WAIT pending completion of I/O (and suspending and resuming tasks), or waiting for storage or disk space to satisfy the request.*
4. *Temporary storage uses the VSAM/BSAM subtask program, DFHVSP, for input/output. (see "VSAM/BSAM Subtasking" on page 89 for more details).*
5. *VSAM communicates with temporary storage control (through task control) when the I/O is complete.*
6. *Temporary storage control analyzes the contents of the RPL to determine the type of completion and any error information.*

Journal Management Function

Journal management provides facilities for creating and managing special-purpose series of sequential data sets (journals) during the execution of CICS and for writing the dynamic log for each transaction that can be backed out. 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 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 macro instructions to request output or retrieval of journal data, or open, close, and position data sets in a journal
- A CICS task for each journal to ensure efficient performance
- One operating system subtask to prevent journal open and close processing, with their possible volume mounting delays, from suspending CICS execution.

For all types of startup, a tape journal is positioned at the beginning of tape, and a disk journal after the latest timestamp.

Journal Management Services Initiated by an Application

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. The prefix data includes a record number, 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 an application program requests immediate output, the operation is initiated as soon as the previous operation in progress ends.

Volume Identification
A tape volume is known by the label:

```
VOLUME NUMBER yyddd/nnn RUN hhmmss
where yyddd   is the date
      nnn     is the volume serial number
      hhmmss  is the start-time.
```

This label is reported to the operator.

Volume Management
This is described in the section "Volume Management" on page 107.

Journal Management Services Performed in Response to Requests from an Application Program

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 so that the user can review or reconstruct changes made in the system.

Journal Management Module (DFHJCP)

Figure 26 on page 105 shows the relationships between the components of journal control.

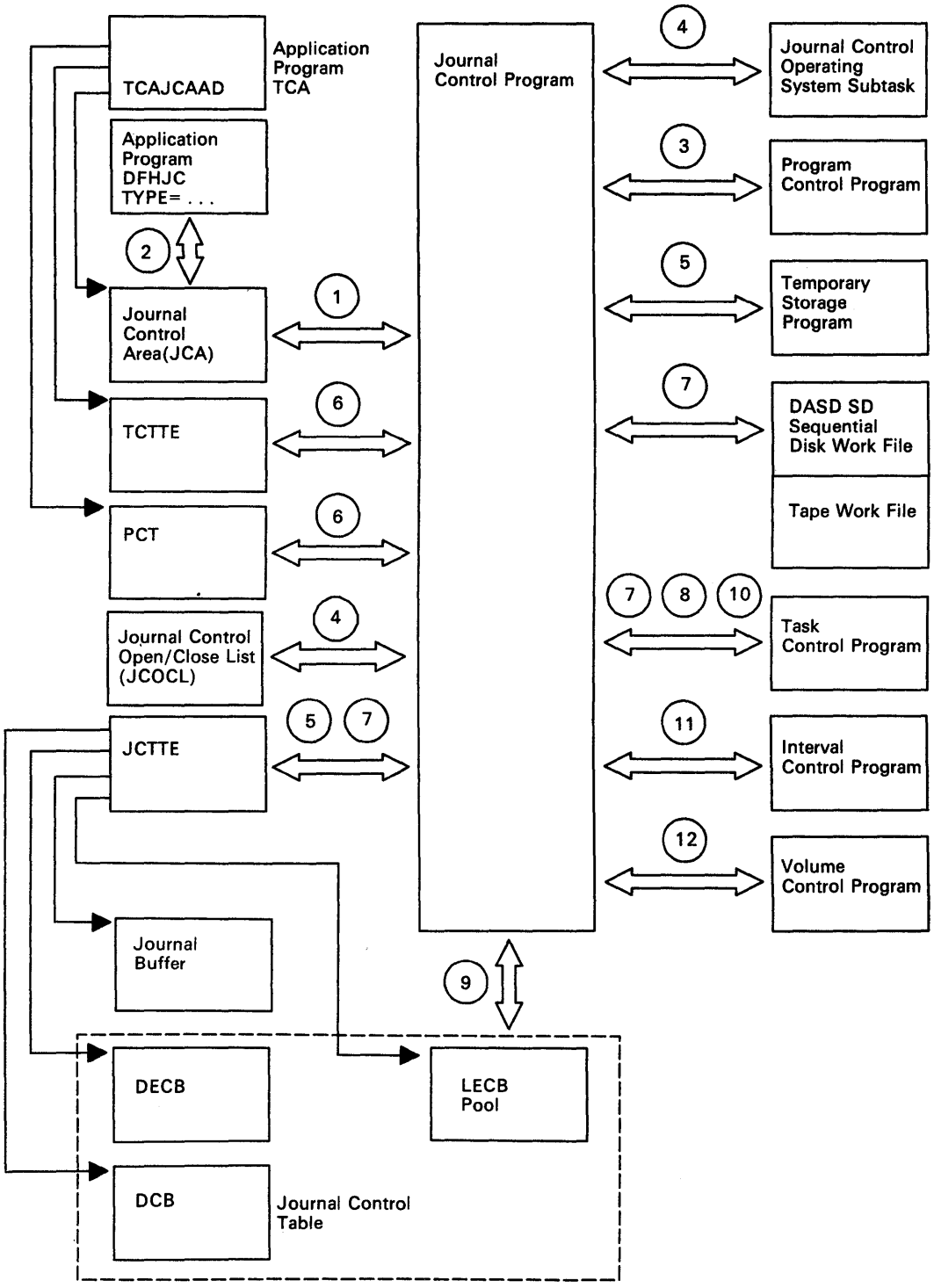


Figure 26. Journal Control Interfaces

Notes:

1. *Journal control commands are communicated to Journal control through a special area, the journal control area (JCA). This area must be acquired for a task by a journal control command before any other journal control command is issued by the task. The TCAJCAAD field of the TCA then points to the JCA for the duration of the task.*

2. *The normal journal control command sets indicators and addresses in the JCA, before branching to journal control.*

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

4. *The journal control open and close transients use the journal control open/close list to communicate requests to an operating system subtask, which then issues the actual open or close. The main CICS task remains dispatchable even while a journal open or close is outstanding. Communication is by the operating system POST plus CICS task control DFHKC TYPE=WAIT. The journal control open/close list is mapped by COPY DFHJCOCL.*

   On an **open-for-output** *request for a disk journal, DFHJCO issues a conditional link to a user-replaceable module, DFHXJCO. This module, if present, is given control just before the open request is passed to the journal subtask. On a close following an* **open-for-output** *request, DFHJCC issues a conditional link to a user-replaceable module, DFHXJCC. This module, if present, is given control just after the close request was completed by the journal subtask.*

5. *If this task requires logging for dynamic backout, a decision is taken whether to log this record or not. If a record is logged, user storage may be required for a dynamic buffer. If the record will not fit into the buffer, the record or the buffer will spill to virtual storage or temporary storage. The queue used to spill to is identified by a byte X'FF' followed by the characters 'DTB' followed by the task number.*

   Output requests cause journal records to be built in the appropriate journal's buffer. Journal control 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 the main journal control program to remain reentrant. The JCTTE is mapped by COPY DFHJCTTE.

   The journal buffer was acquired at system initialization and is owned as transaction storage by a separate, CICS journal task that is normally available for the duration of the CICS system. Journal tasks, like CICS terminal control, 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 journal task for each journal.

6. *While building a journal record, journal control accesses the system area of the caller's TCA for the 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 by COPY DFHJCR.*

7. *Journal control uses the VSAM/BSAM subtask program, DFHVSP, for input/output. (see "VSAM/BSAM Subtasking" on page 89 for more details).*

8. *If the user task wishes to wait (synchronize), journal control 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.*

9. *The journal control table (JCT) contains a pool of LECBs, from which LECBs are allocated and deallocated by journal control as required. The LECB pool is controlled through pointers at the beginning of the JCT, which is mapped by COPY DFHJCTDS.*

10. *When there is no outstanding output I/O event for a journal, journal control issues a DFHKC*

TYPE=SUSPEND macro instruction (another task control macro instruction intended only for system programming) to suspend the journal task.

11. In the particular case when the user task waits on a journal output request for which STARTIO=NO was specified, then interval control is called to provide a timer event control area which will limit the user wait to a maximum of one second.

12. Journal management calls the volume control program (DFHVCP) when opening, closing, or switching standard-labeled tapes.

Volume Management

When standard-labeled tapes are used for journals, volume management is invoked to administer the opening, closing, and switching of journal tapes. Each journal is defined in the CICS system by a chain of volume descriptors (DFHVOLDS). The series definition table (SDT) holds the name, size, pointer to the first volume descriptor, and pointer to the current volume descriptor for each series. The volume descriptor holds the volume ID, a pointer to the next volume descriptor in the series, flags to indicate whether the volume is open or not, free or occupied, and usable or not, a part number, and other information about the volume. The part number is the ordinal position in the journal of the set of data (if any) in the volume. There is also a volume index which holds the volume IDs and the addresses of the corresponding descriptors.

Each tape handled by volume management has a standard header label and a standard trailer label written by the operating system. CICS writes a user label after the header label which gives the volume ID of the preceding tape, and another user label after the trailer label which gives the volume ID of the next tape.

The warm keypoint program (DFHWKP) writes a **restart record** to the restart data set which holds the state of the system log. When the AUTO option is specified on restart, system initialization looks at the restart record to decide what sort of start to perform. If the restart record indicates that the system log was closed, a WARM start is executed. If the restart data set is newly initialized, a COLD start is executed. Otherwise, an emergency restart is executed.

Volume Control Program (DFHVCP)

The volume control program consists of routines for handling the IDs (VOLSERs) of journal tapes. The opening, and closing, of journal tapes is performed by journal control programs. The journal control program and master terminal program invoke DFHVCP by using the DFHVC macro. The following sections describe the operations performed for the possible values of the TYPE option:

ADD

The volume is added as a scratch tape after any other scratch tapes, provided that volume does not exist already.

ADDLIFO

The volume is added as a scratch tape before any other scratch tapes, provided that volume does not exist already.

ADDSERIES

A series name, size, and attributes are added to the series definition table. ADDSERIES is only called during initialization.

BUILDFCB

Data is put in the JFCB so that the volume can be connected to the DCB.

BUILDKP

A keypoint record is created containing significant parts of successive volume descriptors, starting at the given volume or the first volume of the given series.

BUILDNOTE

Given a JCA NOTE, the fields applicable to standard-labeled tapes are given values.

BUILDUL

A user standard header or trailer label for a volume being opened or closed for output is created, with the current time and date.

DELETE

A descriptor is deleted.

GET

The values of various attributes of a volume descriptor are copied to a user-supplied area.

GETSCRATCH

A request is made to the system operator and the tape librarian for the ID of a standard-labeled tape to add as a scratch tape.

INITIATE

The series definition table and the volume index used by volume management are created.

LOCATE

LOCATE is the basic operation of volume management. It is performed as the first operation of most of the other operations. LOCATE returns a volume ID or series name according to the type of request. The request to LOCATE can be for:

The current volume

The volume after the current volume

The volume before the current volume

The first volume in a series

The next volume in a series

The volume most suitable for the next output

The previous volume in a series

The volume specified by a JCA NOTE

The volume referenced by the operating system in a JFCB

The next volume available for output

The current series name

The next series name.

LOCATEWRITABLE

The volume most suitable for output, after the current volume is full, is located. The volume ID is returned. The volume descriptor found is changed to become the first scratch volume in its series.

MAKECURRENT

The specified volume descriptor is made the current volume descriptor in its series.

MERGEKP

Volume descriptors in the specified series are created or updated using information from the restart data set or the system log.

MERGESIP

Create a volume descriptor with the volume ID determined by DFHSIP for use in emergency restart.

MERGEUL

The volume descriptor is updated from the user label of a volume which is being opened or closed for input. If, on emergency restart, the header label indicates a preceding volume, the descriptor for that volume is created if it does not exist already.

READLOGKP

The LOG record is interpreted. The code for this function is in the macro expansion and no call is made to DFHVCP.

SET

New attributes are set in the specified fields of a specified volume descriptor, subject to consistency with existing attributes.

TALLY

Attributes and statistical information for a specified series are returned.

WARN

A specified series is checked, and messages sent to the console and the tape librarian if the number of volumes available as possible successors to the current volume is less than (nominal size – 1).

Sync Point Management Function

Sync point management works in conjunction with other CICS components, such as transient data management, temporary storage management, and file management, to provide the user with the ability to establish points in application programs from which it is convenient to restart. Such a point must be one where all changes to the data base have been committed. (The user can, at any time, backout any uncommitted changes by means of the rollback function.)

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, using the EXEC CICS SYNCPOINT command. Generally, DFHSPP need be invoked by an application program only for a long-running task, 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 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.

DWEs can be used for work to be done before or after the sync point is logged or in the event of transaction backout.

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:

- Scans the chain and passes control to a DWE processor if work is to be done **only**.
- Scans the chain and logs data as required.
- Scans the chain and passes 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 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.

When the DWE is for the task-related user exit, it is processed within DFHSPP. DFHSPP calls DFHEIP to route a **prepare to commit** or **commit** or **backout** to the resource manager according to flags in the DWE. If the resource manager replies **remember**, DFHSPP retains or creates a unit of recovery descriptor (URD).

Task-Related User Exit Resynchronization Function

The purpose of task-related user exit resynchronization is to resolve any in-doubt LUWs. Task-related user exit resynchronization is called by sync point management during execution of the RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

Task-related user exit resynchronization is performed by the module DFHRMSY.

DFHSPP processes the EXEC CICS RESYNC command. It scans URDs for the name of the resource manager, and calls DFHRMSY with arguments which are passed to the task-related user exit interface, which creates deferred work elements (DWEs) and transaction interface elements (TIEs) with the appropriate flags for **commit**, or **backout,** or **lost-to-cold-start**. **Lost-to-cold-start** means that the recovery token in the URD is older than the time of the last CICS cold start.

Chapter 2.2. System Services Component

A number of ancillary application programs are included in CICS 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
- Security interface program
- Resource definition online
- Master terminal program
- Supervisory terminal
- Operator terminal
- System statistics
- Dynamic open/close
- Time-of-day control
- Field engineering program
- Message switching
- Dynamic allocation sample program
- System spooling interface.

Sign-On/Sign-Off

The sign-on/sign-off function is optional. If selected, it can be used in various ways. 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. The security key is matched with the value in the PCT or the resource list.

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 clears the information that was set in the TCTTE by sign-on, and frees the SNTTE chained from the TCTTE by sign-on. In signing off, the operator can optionally request that the terminal be logically disconnected from CICS. 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.

For terminals supported by VTAM, CLSDST is executed if CSSF has the LOGOFF or the GOODNIGHT option.

Security Program

CICS provides an optional interface for greater checking of user, transaction, and resource authorization by means of the DFHXSP module. This interface can be to the MVS licensed program Resource Access Control Facility (RACF). Instead of RACF, a user-written program can perform the

check. The user-written program can replace DFHXSP or DFHXSE (which is a module called by DFHXSP). To activate the external security interface, entries must be made in the PCT, the SIT, and the SNT.

The link between CICS and the security manager is the module DFHXSE which is invoked by the macro DFHSEC. If needed, RACF is invoked by a call to the CICS SVC routine, DFHCSVC, and subsequently to DFHXSS. DFHSEC has parameters for sign-on, sign-off, authorization check, initialization, creating a DSECT, and creating a LIFO parameter area. Any program which satisfies the entry and exit conventions given in the *CICS/MVS Customization Guide* may be substituted for DFHXSP. The security program is initialized by DFHSII1.

During CICS initialization, the RACF macro RACROUTE REQUEST=LIST is issued to build in storage profiles for the CICS transaction class and the PSB class. Any failure in RACROUTE causes an abnormal return code, which is displayed by the message DFH1567 and allows the console operator to continue without security, to retry, or to cancel CICS.

At sign-on, the operator name is searched for in the SNT. If security is specified and the name cannot be found, the DEFAULT entry, if present in the SNT, is used. If the name is found, a pseudo SNT entry is created and addressed from the TCTTE, and the user is prompted for the current password, a new password, and/or an OPID card, if required. A null input cancels the sign-on. If an OPID card is presented without sign-on, the information is saved and the operator prompted for sign-on. The RACF macro RACROUTE REQUEST=VERIFY creates the access control environment element (ACEE), whose address is held in the SNNT after sign-on.

At sign-off, if security was used, the RACF macro RACROUTE REQUEST=VERIFY with the DELETE option is issued to free the ACEE created by sign-on.

When a transaction is about to be dispatched, DFHKCP calls DFHZSUP which does a security check using the RACF macro RACROUTE REQUEST=FASTAUTH before starting the transaction. If security is breached, then DFHZSUP passes control to DFHACP with an abend code in TCAPCABR.

When a transaction attempts to schedule a PSB, DL/I invokes the security manager to check that the user is authorized to use the PSB.

The CICS modules that call the security program issue messages about who, when, where, and with what result an operator attempted to sign-on or sign-off. These messages use DFHMGP to send data to the console or to the transient data destination CSCS.

RACF protection can be given to certain tables. Each table has one byte whose value is a position in a 24-byte field in the PCT. If a bit is set on in the PCT, and the corresponding bit is set on in the user's TCTTE, then the operator can have access to that resource.

Resource Definition Online

The CEDA transaction creates and updates the CICS system definition (CSD) file. The CSD file is a VSAM KSDS cluster. The CEDA transaction receives a command, and invokes DFHEDAD and DFHEDAP which pass the command to the allocation management program, DFHAMP.

DFHAMP analyzes the commands, and calls the definition file management program, DFHDMP, to process changes to records in the CSD. For the INSTALL command, DFHAMP also calls DFHKCP, DFHPCP, DFHSPP, and DFHZCQ.

These programs also handle the CEDB transaction, which does the same as CEDA except for INSTALL GROUP, and the CEDC transaction, which can only interrogate the CSD file.

The CSD utility program, DFHCSDUP, is an offline program which performs functions similar to the CEDA transaction.

When CICS determines that the NETNAME for a logon request is unknown and autoinstallation of terminals is available on the system, it defines the terminal dynamically using the information supplied by the CINIT request unit. For a description of how the CEDA transaction handles terminal resources, see "Defining Terminals to CICS" on page 82.

Master Terminal Program

The master terminal program (DFHMTP) is an optional feature of CICS selected at system initialization. This program consists of seven modules: DFHMTPA through to DFHMTPG. It is a service program that provides the user with the means of dynamically changing certain system parameters, the status of lines, control units or terminals.

The master terminal program is invoked by operator keying of the proper transaction identification at a master terminal, a supervisory terminal or an ordinary terminal. The transaction identification can 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.

Figure 27 on page 114 shows the master terminal program interfaces.

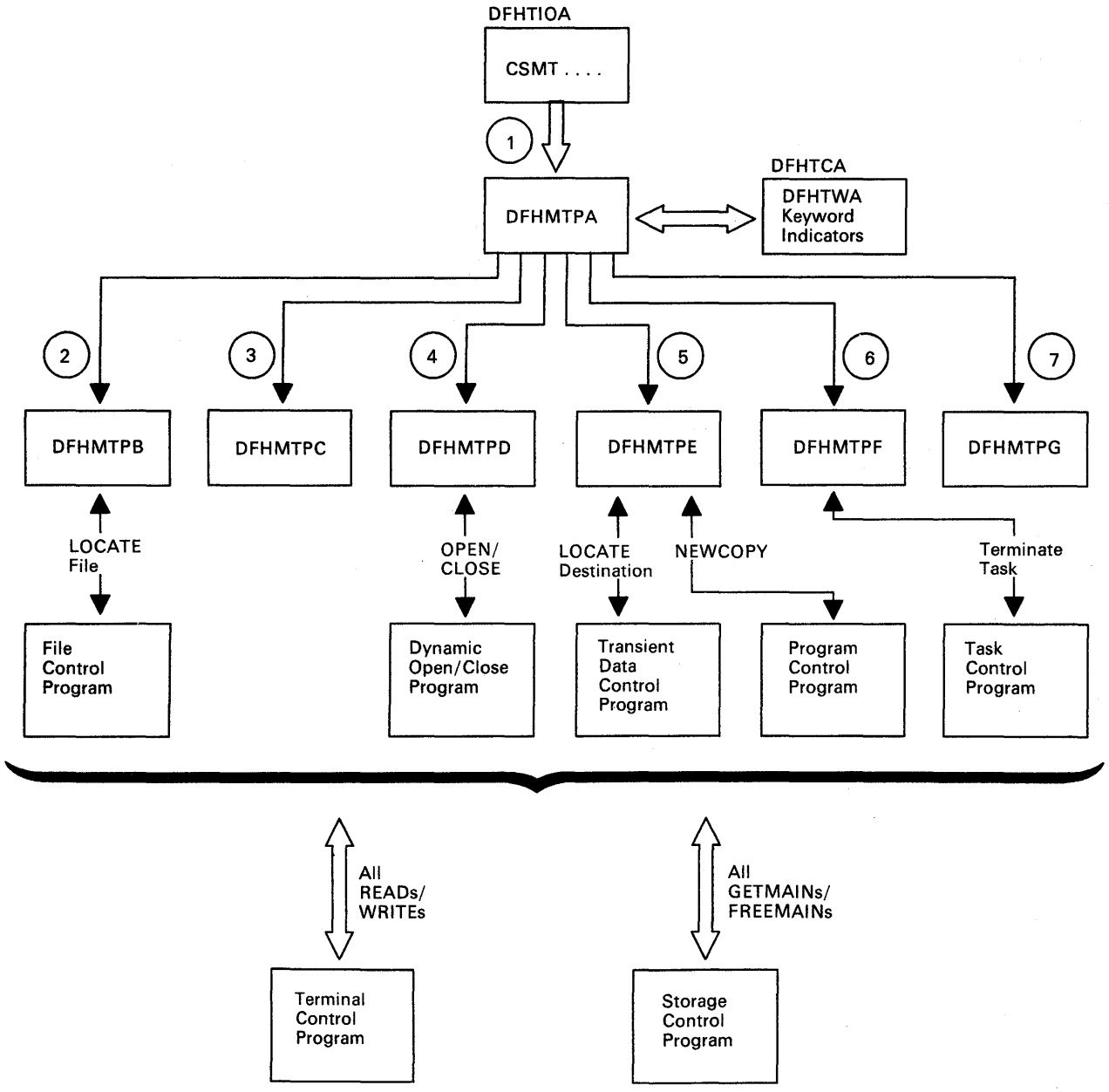


Figure 27. Master Terminal Program Interfaces

Notes:

1. *All transaction input under transaction identification CSMT, CSST, or CSOT is placed in a TIOA accessible to DFHMTPA. The input is scanned to determine whether sufficient keywords are present, to fully specify the requested service. Keyword indicators are set in the requesting task's TWA for all keywords entered. Control is transferred to the DFHMTP module that provides the requested service. If time or runaway task services are requested, DFHMTPA responds to the request. Otherwise, additional information is requested from the input terminal as needed. DFHMTPA interfaces with terminal control for reads and writes and with storage control to obtain necessary storage, as do all other DFHMTP modules.*

2. *DFHMTPB responds to all requests for file, cushion, maximum task, negative poll delay and trace services. The file control program (DFHFCP) is used to locate file entries in the file control table (FCT).*

3. *DFHMTPC responds to all requests for terminal services.*

4. *DFHMTPD responds to all requests for open, close, and switch of dump data sets. The dynamic open/close program (DFHOCP) is used to perform the actual file open or close.*

5. *DFHMTPE responds to all requests for transient data destinations, stall, trigger level and copy services. The transient data control program (DFHTDP) is used to locate destinations in the destination control table (DCT). Program control (DFHPCP) services are requested by means of a DFHPC TYPE=BLDL macro instruction when responding to a NEWCOPY request.*

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

7. *DFHMTPG responds to all requests for transaction and program services.*

8. *DFHSTP responds to requests for system shutdown.*

Enhanced Master Terminal

The enhanced master terminal program (DFHEMTP and DFHEMTD) is a CICS-supplied transaction that allows master terminal commands to be syntax-checked and executed. Transaction IDs CEMT, CEST, and CEOT are used instead of CSMT, CSST, and CSOT respectively. The syntax analysis of the commands is table driven in the same way as the command interpreter.

For command execution, an argument list is built and the EXEC interface program is invoked. For a master terminal command, this passes control to one of the following modules. The modules and the commands they process are:

DFHEMA	Task values (for example, MAXTASKS)
	TCLASS
	BATCH
	RESET
DFHEMB	VTAM
	DUMP
	IRC
	IRBATCH
	AUXTRACE
	TRACE
	PITRACE
	SNAP
	SHUTDOWN
DFHEMC	TERMINAL
	NETNAME
DFHEMD	CONNECTION
	MODENAME
DFHEME	TRANSACTION
	TASK
DFHEMF	PROGRAM
	QUEUE
DFHEMG	CONTROL
	LINE
DFHEMH	VOLUME
	JOURNAL
DFHEMI	DATASET
	DLIDATABASE
	RECONNECT

Supervisory 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 CEST or CSST, supervisors can change the service status or the processing status 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 only. Supervisor requests are checked for validity and processed by the master terminal program.

Operator Terminal

Terminal operators who are neither supervisory operators nor the master terminal operator are only able to control the service status and the processing status of their own terminals, using the command CEOT or CSOT. Operator terminal requests are checked for validity and processed by the master terminal program.

System Statistics

System statistics are maintained by CICS management programs during the execution of CICS and can be made available **on request**, or **automatically** at intervals, by any operator whose security code allows access to such information. Additionally, system statistics are produced on normal termination of the system.

Requested system statistics are available in full, or in part, at a user-defined destination as variable-length, unblocked records with a maximum block size of 120 (decimal). (The default destination is CSSL.) The operator can optionally request that all selected statistics (except total number of tasks) be reset to zero after each output. Requested system statistics cannot be obtained while automatic system statistics are being recorded.

Automatic system statistics are recorded in full at user-specified intervals (for example every two minutes). Two destinations (CSSM and CSSN) are provided for automatic statistics; they allow the recording of statistics on one destination in parallel with offline processing of statistics previously recorded on the other destination. (See the *CICS/MVS Operations Guide* for further information.) Automatic statistics are cumulative only for the period specified; that is, the statistics are reset to zero at the end of each period.

Note: It is advisable to use automatic statistics to avoid overflow of the statistics counters. Use the CSTT transaction to specify when statistics are to be written out.

Termination statistics sent to CSSL on normal shutdown have the same format as requested statistics. Statistics produced at system termination do **not** include the data that was produced (and sent to CSSM and CSSN) during **automatic** statistics recording; but they **do** include all data that has accumulated since the totals were last set to zero.

Dynamic Open/Close Program

The dynamic open/close program provides open/close capabilities for dump data sets, transient data extrapartition data sets, and file control data sets. These facilities can be invoked by the CICS master terminal program or through the use of DFHOC macro instructions in an assembler-language application program.

Figure 28 on page 118 shows the dynamic open/close program interfaces.

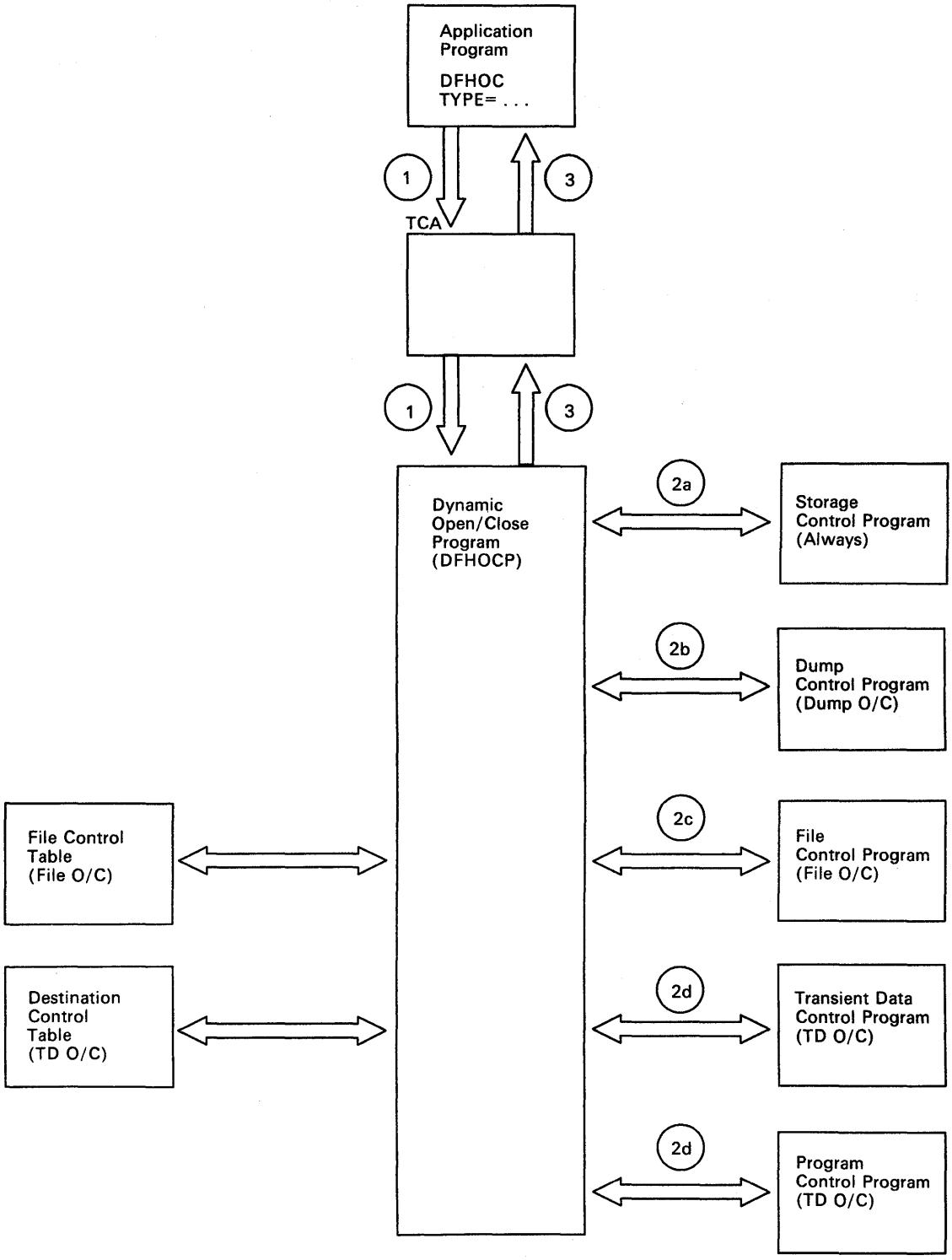


Figure 28. Dynamic Open/Close Program Interfaces

Notes:

1. *The DFHOC macro expansion issues a program control DFHPC TYPE=LINK macro instruction to transfer control to the dynamic open/close program. This transfer is not apparent to the program issuing the DFHOC macro instruction.*

2. *Dynamic open/close communicates with:*

   a. *Storage control, in response to all requests for services*

   b. *Dump control, in response to requests to open, close, or switch of dump data sets*

   c. *File control to open or close file control data sets*

   d. *Transient data control and program control to open or close extrapartition data sets.*

3. *Return to the requesting program is achieved by means of a program control DFHPC TYPE=RETURN macro instruction, which is not apparent to the requesting program.*

Time-of-Day Control

The optional time-of-day control program (DFHTAJP) makes it possible for the user to operate CICS continuously, for more than 24 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 recognizes the situation where a negative change in the time of day has occurred and adjusts the expiration times maintained by CICS, 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.

Field Engineering Program

The field engineering program (DFHFEP) is a CICS system service function primarily designed for an IBM field engineer to use when installing new terminals. When CICS 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.

This program performs all requests made by the CSFE transaction. CSFE can request the activation or deactivation of:

- Certain kinds of trace (for example, user or FE trace)
- Storage freeze
- Storage violation trap
- Global trap/trace exit.

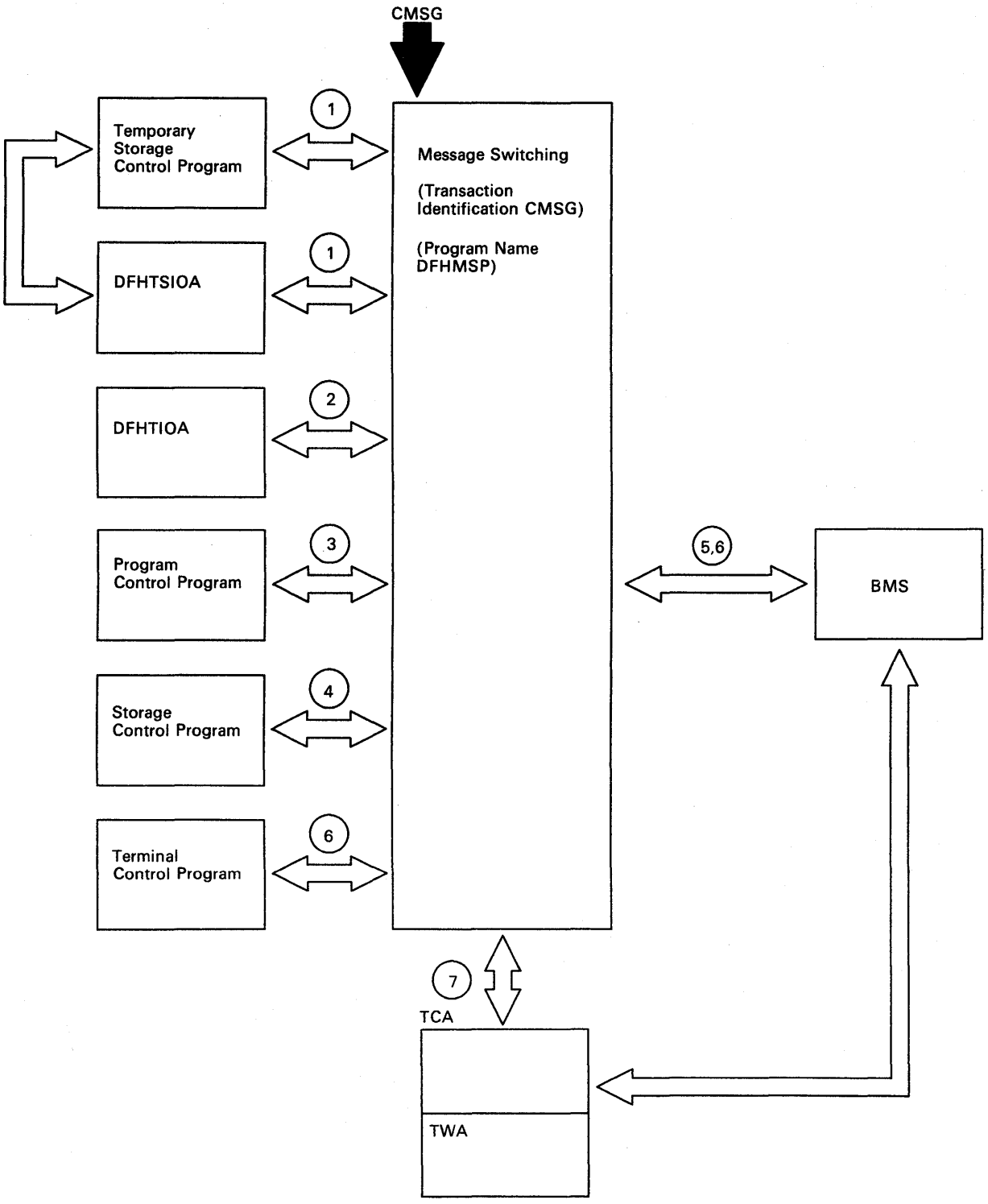


Figure 29. Message Switching Interfaces

Message Switching

Message switching runs as a task under CICS. A terminal operator requests activation of this task by entry of the transaction identification CMSG (or another installation-defined 4-character transaction identification), followed by appropriate parameters. Once initiated, message switching interfaces with CICS basic mapping support (BMS) and CICS management functions. Although message switching appears conversational to the terminal operator, the message switching task is terminated with each terminal response. 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. Actually, XXXX is dynamically moved into TCANXTID prior to issuing DFHPC TYPE=RETURN.

Figure 29 on page 120 shows the relationships between the components of message switching.

Notes:

1. *If the first 4 characters of the TIOA (not including a possible set buffer address (SBA) sequence from an IBM 3270 Information Display System) 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 is also a force continuation. In such a case, information has been stored in and has to be retrieved from temporary storage (using a record key of 1-byte X'FC', 4-byte terminal identification, and 3-byte C'MSG') to allow the task to resume where it left off.*

2. *The operands in the input TIOA are processed and their values and status are stored in the TWA.*

3. *If a ROUTE operand specifies terminal list table(s) (TLTs) for a standard routing list, a DFHPC TYPE=LOAD macro instruction is issued to request the Program Control program to load the requested TLT(s).*

4. *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.*

5. *Message switching requests BMS routing functions by means of 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.*

6. *Message switching interfaces with BMS using DFHBMS TYPE=(EDIT,OUT) and with CICS terminal control using DFHTC TYPE=WRITE for the IBM 3270 Information Display System only, in providing responses to terminals. These can indicate normal completion, signal that input is to continue, or provide notification of input error.*

7. *Like any other task, message switching has a task control area (TCA) in which values may be placed prior to issuing CICS 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.*

Dynamic Allocation Sample Program (IBM 3270 Only)

The program runs as a CICS transaction, using CICS function at the command level wherever possible. It does not modify any CICS control blocks. Only the DYNALLOC function is available through the program; any manipulation of the environment before or after the DYNALLOC request must be done by other means.

The flow in a normal invocation is as follows. The main program, DFH99M receives control from CICS, and carries out initialization. This includes determining the screen size and allocating input and output buffer sections, and issuing initial messages. It then invokes DFH99GI to get the input command from the terminal. Upon return, if the command was null, the main program terminates, issuing a final message.

The command obtained has its start and end addresses stored in the global communications area, COMM. The main program allocates storage for tokenized text, and calls DFH99TK to tokenize the command. If errors were detected at this stage, further analysis of the command is bypassed.

Following successful tokenizing, the main program calls DFH99FP to analyze the verb keyword. DFH99FP calls DFH99LK to look up the verb keyword in the table, DFH99T. DFH99LK calls DFH99MT if an abbreviation is possible. Upon finding the matching verb, DFH99FP puts the address of the operand section of the table into COMM, and puts the function code into the DYNALLOC request block.

The main program now calls DFH99KO to process the operand keywords. Each keyword in turn is looked up in the table by calling DFH99LK, and the value coded for the keyword is checked against the attributes in the table. DFH99KO then starts off a text unit with the appropriate code, and, depending on the attributes the value should have, calls a convert routine.

For character and numeric strings, DFH99CC is called. It validates the string, and puts its length and value into the text unit.

For binary variables, DFH99BC is called. It validates the value, converts it to binary of the required length, and puts its length and value into the text unit.

For keyword values, DFH99KC is called. It looks up the value in the description part of the keyword table using DFH99LK, and puts the coded equivalent value and its length into the text unit.

When a keyword specifying a returned value is encountered, DHF99KO makes an entry on the returned value chain, which is anchored in COMM. This addresses the keyword entry in DFH99T, the text unit where the value will be returned, and the next entry. In this case the convert routine is still called, but it only reserves storage in the text unit, setting the length to the maximum and the value to zeros.

When all the operand keywords have been processed, DFH99KO returns to the main program, which calls DFH99DY to issue the DYNALLOC request.

DFH99DY sets up the remaining parts of the parameter list, and if no errors too severe have been detected, a subtask is ATTACHed to issue the DYNALLOC SVC. A WAIT EVENT is then issued against the subtask termination ECB. When the subtask ends, and CICS dispatches the program again, the DYNALLOC return code is captured from the subtask ECB, and the error and reason codes from the DYNALLOC request block, and a message is issued to give these values to the terminal.

DFH99DY then returns to the main program, which calls DFH99RP to process returned values. DFH99RP scans the returned value chain, and for each element issues a message containing the keyword and the value found in the text unit. If a returned value corresponds to a keyword value, DFH99KR is called to look up the value in the description, and issue the message.

Processing of the command is now complete, and the main program reinitializes for the next one, and loops back to the point where it calls DFH99GI.

Messages are issued at many places, using macros. The macro expansion ends with a call to DFH99MP, which ensures that a new line is started for each new message, and calls DFH99ML, the message editor. Input to the message editor is a list of tokens, and each one is picked up in turn

and converted to displayable text. For each piece of text, DFH99TX is called, which inserts the text into the output buffer, starting a new line if necessary. This arranges that a word is never split over two lines.

At the end of processing the command, the main program calls DFH99MP with no parameters, which causes it to send the output buffer to the terminal, and initialize it to empty.

System Spooling Interface

The system spooling interface program opens a system spooling file for either input or output, reads or writes a file, and closes a file. These functions are for system programmer use. The input is single-threaded, so only one transaction can use it at a time. You should try to avoid one transaction excluding other transactions by, for example:

- Not executing a SPOOLOPEN command immediately after a SPOOLCLOSE command.
- Not executing a SPOOLOPEN command until a file can be processed completely.
- Issuing a SPOOLCLOSE command as soon as possible after a SPOOLREAD command.
- Using different external writer names, or different classes within names.

An application can send files to a remote location by specifying the NODE of the location, and the USERID (or external writer name) of the user at that location. When you wish to retrieve a file at the remote location, specify the external writer name, and you can then retrieve reports from that writer. For security reasons, the external writer name must begin with the same four characters as the CICS APPLID. The remote system to which a file or report is sent, or from which it is received, must have JES under MVS, or VM.

System Spooling Interface Modules

The SPOOLOPEN command dynamically allocates input or output files using the CICS SVC, and an ACB is opened to process the file. For an input file, the IEFSSREQ macro is also issued to determine which file to process. The SPOOLREAD or SPOOLWRITE commands cause GETs or PUTs to be issued using the ACB. The SPOOLCLOSE command dynamically deallocates a file, and causes it to be either transmitted or deleted. All processing which could cause CICS to be suspended is performed under an operating system subtask which is initiated by subtask management, DFHSKP. DFHPSPST runs under CICS, but DFHPSPSS, and modules called as a result, run under the subtask.

Normal Flow

When a system spooling interface command is executed, the normal sequence of invocation of modules is:

1. DFHEIP
2. DFHEPS
3. DFHPSP
4. DFHPSPSS
5. DFHPSPST
6. DFHCSSVC
7. DFHPSSVC.

Abnormal Flow

If a user transaction terminates without issuing a SPOOLCLOSE command, DFHPSPDW is invoked to process a deferred work element (DWE) that was set up when the SPOOLOPEN command was processed. This closes the file in the usual way.

Chapter 2.3. System Monitoring Component

The system monitoring component consists of the following functions that provide diagnostic traces and dumps.

- Trace management
- Dump management
- Formatted dump program
- Trace utility program
- Dump utility program
- CICS monitoring facility.

Trace Management (DFHTRP)

The trace control program (DFHTRP) is designed as a debugging aid for the programmer. It provides an easy and convenient means of tracing system activity. (The content and format of entries in the trace table are explained in detail in the *CICS/MVS Problem Determination Guide*.)

Figure 30 shows the relationships between the components of trace management.

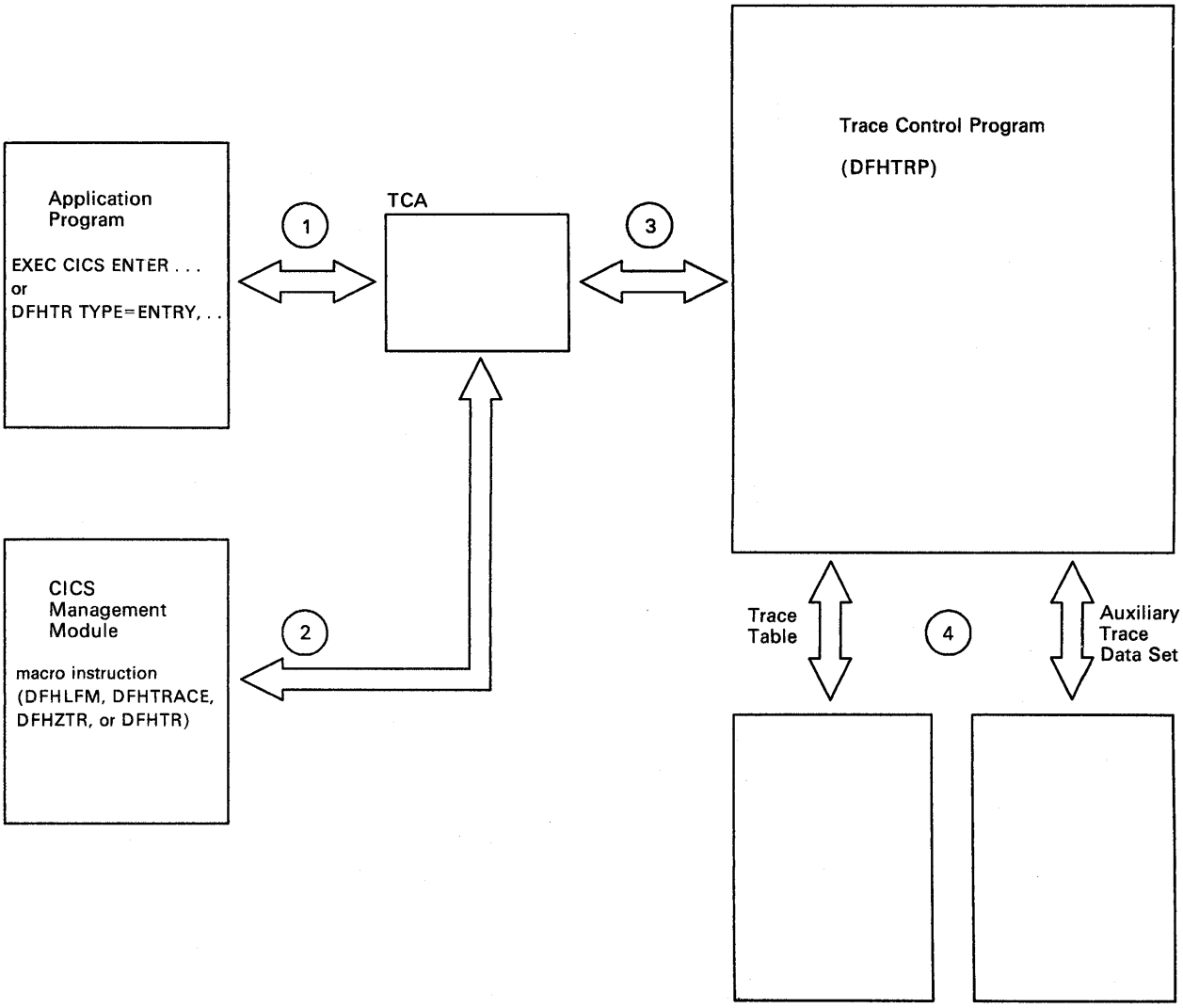


Figure 30. Trace Control Interfaces

Notes:

1. *The application program can issue EXEC CICS ENTER commands or DFHTR macro instructions to cause user-defined entries to be recorded in the trace table.*

2. *Standard CICS entries are recorded in the trace table whenever CICS commands or macro instructions of certain types are issued by an application program or a CICS system program.*

   These entries in the trace table are generated by special macro instructions (DFHLFM, DFHTRACE, DFHZTR, or DFHTR) that are included in the CICS management modules.

3. *Control is returned to the requesting program.*

4. *When the auxiliary trace facility is active, trace table entries are also written to the auxiliary trace data set, which can be printed offline using the trace utility program (DFHTUP).*

Auxiliary Trace Management

When the auxiliary trace facility is active, and main-storage trace is also active, trace entries are written to a sequential data set (tape or disk) for subsequent offline processing by the trace utility program (DFHTUP). The format of trace entries in the auxiliary trace data set is identical to that in the trace table.

In addition, auxiliary trace output contains some X'D0' trace entries for task-related activities that do not feature in the main-storage trace unless auxiliary trace is active. These are task dispatch, task suspend, system wait, and system resume.

Information in the auxiliary trace data set can be used for problem determination and performance tuning. Auxiliary trace has the advantage that required trace entries are not overwritten, as they can be in the main-storage trace table, which is used in a wraparound manner.

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 dump utility program. Requests for storage dumps are communicated to the dump management by EXEC CICS DUMP commands or by DFHDC macro instructions.

Dump management records dumps of transactions on sequential data sets on either magnetic tape or direct-access storage. CICS allows you to open or close the active dump data set during CICS execution.

Optionally, the user can define two dump data sets (DFHDMPA and DFHDMPB), alternating between them during the execution of CICS. If DFHDMPA is opened by system initialization and if a request is issued, through the CEMT or 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, overwriting previous dumps.

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 be printed later by the dump utility program. DFHDCP can be called by either an application program or a CICS 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.

Figure 31 on page 128 shows the relationships between the components of dump management.

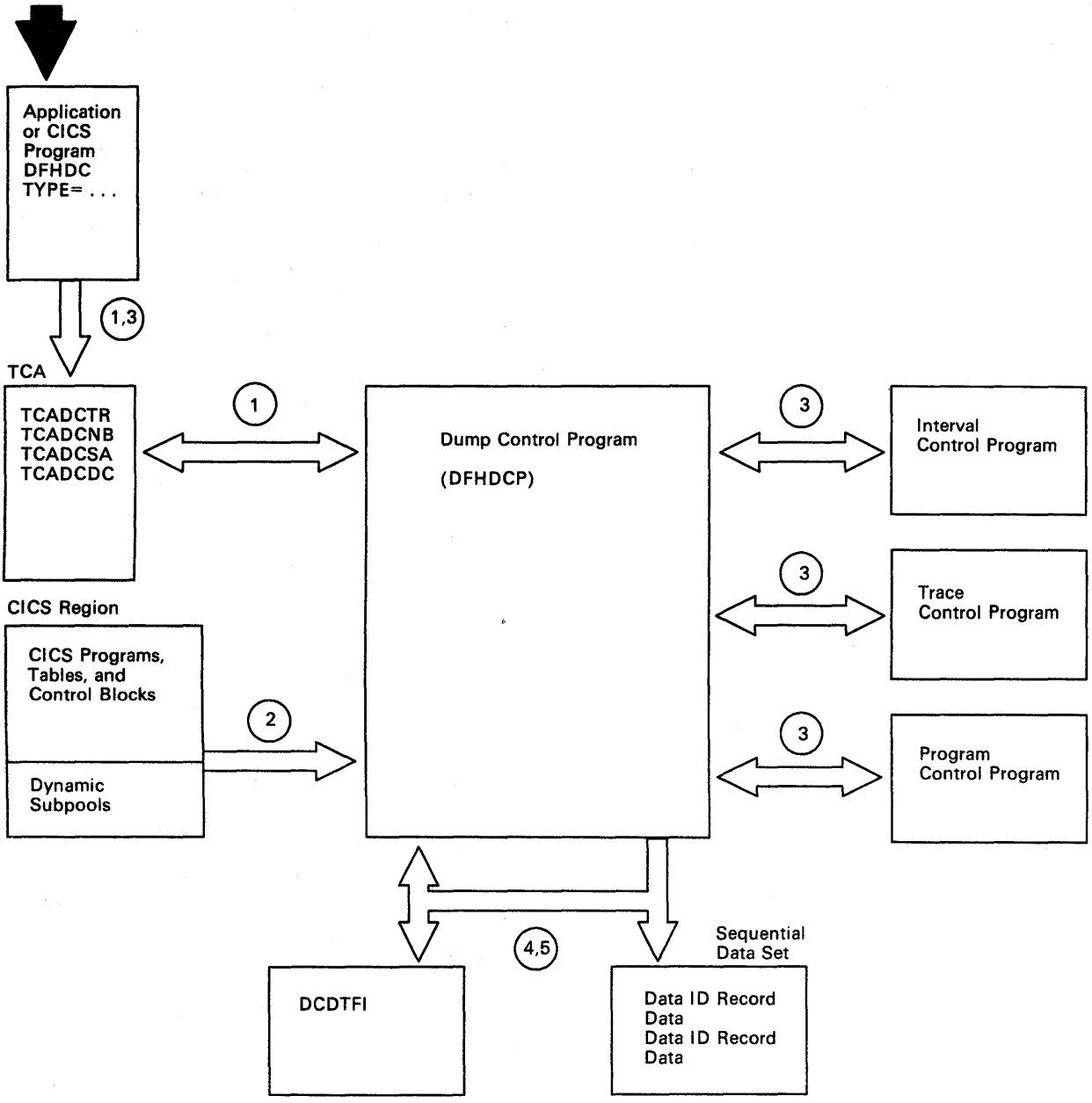


Figure 31. Dump Control Interfaces

Notes:

1. *DFHDCP can be called by either an application program or a CICS management program to write out the contents of main storage at any time. The dump command or DFHDC macro instruction sets the type of request switch (TCADCTR), number of bytes to be dumped (TCADCNB), storage address (TCADCSA) and dump identifier (TCADCDC) in the requesting program's TCA.*

2. *The common system area (CSA) and task control area (TCA) are always the first areas to be dumped. Additional areas that may be dumped include: transaction storage, trace table, terminal storage, program and register save areas, and system control tables.*

3. *DFHDCP communicates with interval control, trace control, and program control to set runaway task control, request and release a lock of the dump facility, request a trace, and reset and test runaway task control.*

4. *Dump control refers to DCDTFI in an I/O control block in DFHDCP's static storage, which identifies a dummy ECB referenced in the lock and release of the dump facility.*

5. *Output is written to a sequential data set on tape or disk. Records have a variable blocked format. A single record may be a CICS system control table, one TCA, or an entire program. Each logical record contains a 1-byte identifier to indicate the type of control block, table, or program that it contains.*

Formatted Dump Program

The formatted dump program (DFHFDP) is invoked by abnormal termination of CICS, by specification of DUMP at CICS shutdown time, or by the issue of a CEMT or CSMT SNAP command by the master terminal operator. It is also optionally invoked on a program check, storage violation, or system abend.

The output of the formatted dump program can be in one of three forms, depending on the specification made either at CICS startup using one of the start parameters, or at system initialization by the DUMP parameter of the DFHSIT macro. The three forms of output thus specified are:

- PARTN – a dump of the CICS address space only.
- FORMAT – a listing of a series of CICS control blocks, each dumped in as logical an order as possible. Fields to be highlighted in each control block follow the hexadecimal dump of the appropriate control block.
- FULL – both the above areas.

The first line of the dump is the short symptom string.

DUMP=(PARTN,SNAP) results in an MVS SNAP macro being issued. This dump is written to ddname DFHSNAP, usually defined as the SYSOUT=A printer.
DUMP=(PARTN,SDUMP) results in an MVS SDUMP macro being issued. This dump is written to a SYS1.DUMP data set.

The formatted dump program consists of the following modules: DFHFDP, DFHFDB, and DFHFDC.

DFHFDP is the control module which acts as an interface module to CICS 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.

The main routines and subroutines of DFHFDP are:

- Initialization (at CICS startup)
- Main dump routine
- Subroutines 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 that executes the text contained in DFHFDC. It calls DFHFDP 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.
- Priming 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 scanning – 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 – fetching the operation code of the next descriptor and branches to the appropriate descriptor processing routine.
- Processing descriptor routines.
- Processing termination routines.
- Processing error 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, etcetera.

Trace Utility Program

The trace utility program (DFHTUP) formats and prints the contents of the auxiliary trace data set. This program operates in batch mode, and formats each trace entry to show the CICS component that made the entry and the function being performed. The display format is the same as that used for the interpreted display of the trace table in a CICS dump. When using the trace utility program, not all entries need be printed; the entries to be printed can be selected by transaction identifier, task identifier, terminal identifier, trace identifier, or time interval.

Dump Utility Program

The output from dump management, generated during the execution of CICS, 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.

CICS Monitoring Facility

The CICS monitoring facility records operational data concerning CICS execution. The data can be divided into three classes:

- Accounting class – including transaction ID, terminal ID and operator ID
- Performance class – for example, time taken, storage used
- Exception class – records, for example, exceptional conditions such as string waits.

Monitoring can be switched on or off by a master terminal CSTT command which uses the DFHCMON module. Monitoring is invoked by the DFHTR, DFHTRACE, DFHLFM, or DFHEMP macros at the appropriate points in CICS or user programs. These macros call DFHCMP, which collects and records the data by interpreting the monitor control table (MCT) for user data and a table internal to DFHCMP for system data. Each record is time-stamped with the time obtained using the store clock (STCK) instruction.

The MCT contains the definition of user event monitoring points (EMPs), and defines when and how the user fields in the accounting or performance class records are to be manipulated. The size of the user area is calculated from the TYPE=EMP macro.

The MCT specifies how the data is to be recorded, and where the data is to be written. Output is to a journal data set, or a system monitoring facility (SMF) data set. The particular journal is specified in the MCT.

The data on the journal is enclosed in SMF record envelopes. The data is self-defining with respect to a dictionary that is also in the journal. The dictionary is written to the data set when a class of monitoring is enabled, and, subsequently, on volume switches.

The data for each user task is built in the TCA in an area between the TWA and LIFO. The area obtained is correct for the levels of monitoring active when the task is attached.

The format of all CICS monitoring facility records is the SMF Type 110 record; the label record in each block corresponds to the SMF record header. Each journal record for the different monitoring classes has added header information. For MVS/SE2 and later systems, the monitoring data is passed directly to SMF for output to a standard SMF data set if JTYPE=SMF is specified in the JCT.

DFHCCMF invokes monitoring to flush relevant classes of information into data sets at specified time intervals. DFHCCMF issues a DFHIC macro to activate itself at the next time interval.

The CSTT transaction invokes DFHSTKC which links to DFHCMON if the MONITOR operand is specified. DFHCMON starts and stops the different classes of monitoring, and checks that the environment is suitable for starting the specified classes. For performance class monitoring, DFHCMON combines the dictionaries for CICS-system and user-defined objects. For this class, it also deduces, from a map in the monitoring control table, which fields to include or exclude. DFHCMON ensures, by using a DFHSC GETMAIN if necessary, that sufficient storage is available for CICS system tasks.

User tasks which were active before a particular class of monitoring was switched on cannot be monitored by that class as no extra storage is available. DFHCMON is called by the initialization program, DFHSII1, to switch on individual monitoring classes as specified by the MONITOR keyword in the SIT. When further classes of monitoring are enabled for CICS system tasks, storage is obtained from the shared subpool. This storage is used to block data before being sent to the journal data set.

DFHCMON also produces dictionaries at the beginning of each new journal and puts them in storage chains which are anchored from the journal control table table entry (JCTTE).

DFHCMP interprets entries in the MCT to extract and record the monitoring information.

User Exits

A user exit can take place in DFHCMP after completing a monitoring record and before moving the record to a buffer. The user exit applies to the accounting, performance, and exception classes of record.

Instead of a global user exit, a task related user exit is used. On entry to the exit program, register 1 addresses the user exit parameter list (defined by DFHUEXIT TYPE=RM). In this parameter list, UEPHMSA points to a register save area. In the register save area, the field where register 1 is saved points to the CICS monitoring facility parameter list. This parameter list is allocated on the first exit for any task. When an exit in invoked for the first time for any task, a transaction interface element (TIE) is created to exchange information with the task-related user exit interface.

Chapter 2.4. System Reliability Component

The system reliability component consists of programs that perform functions to help the user deal with error conditions. The functions are performed online.

- System recovery management program
- Dynamic backout program
- Retry program
- Abnormal condition program
- Program error program
- Terminal abnormal condition program (BTAM, GAM)
- Terminal error program (BTAM, GAM)
- Node abnormal condition program (VTAM)
- Node error program (VTAM)
- Emergency restart
- Keypoint program
- Task-related user exit recovery.

System Recovery Management Program

The system recovery program (DFHSRP) is a generalized abnormal termination handler that receives control from the operating system when a program check or abend condition is recognized. It provides program interrupt logic for the capture and recovery of program-check interrupts and system abends. A program check interrupt normally causes DFHSRP to invoke a DFHPC ABEND of the task in which it arose. A system abend condition may be handled by user code or IBM-supplied code, after which CICS will attempt to continue to run.

Figure 32 shows the relationships between the components of system recovery.

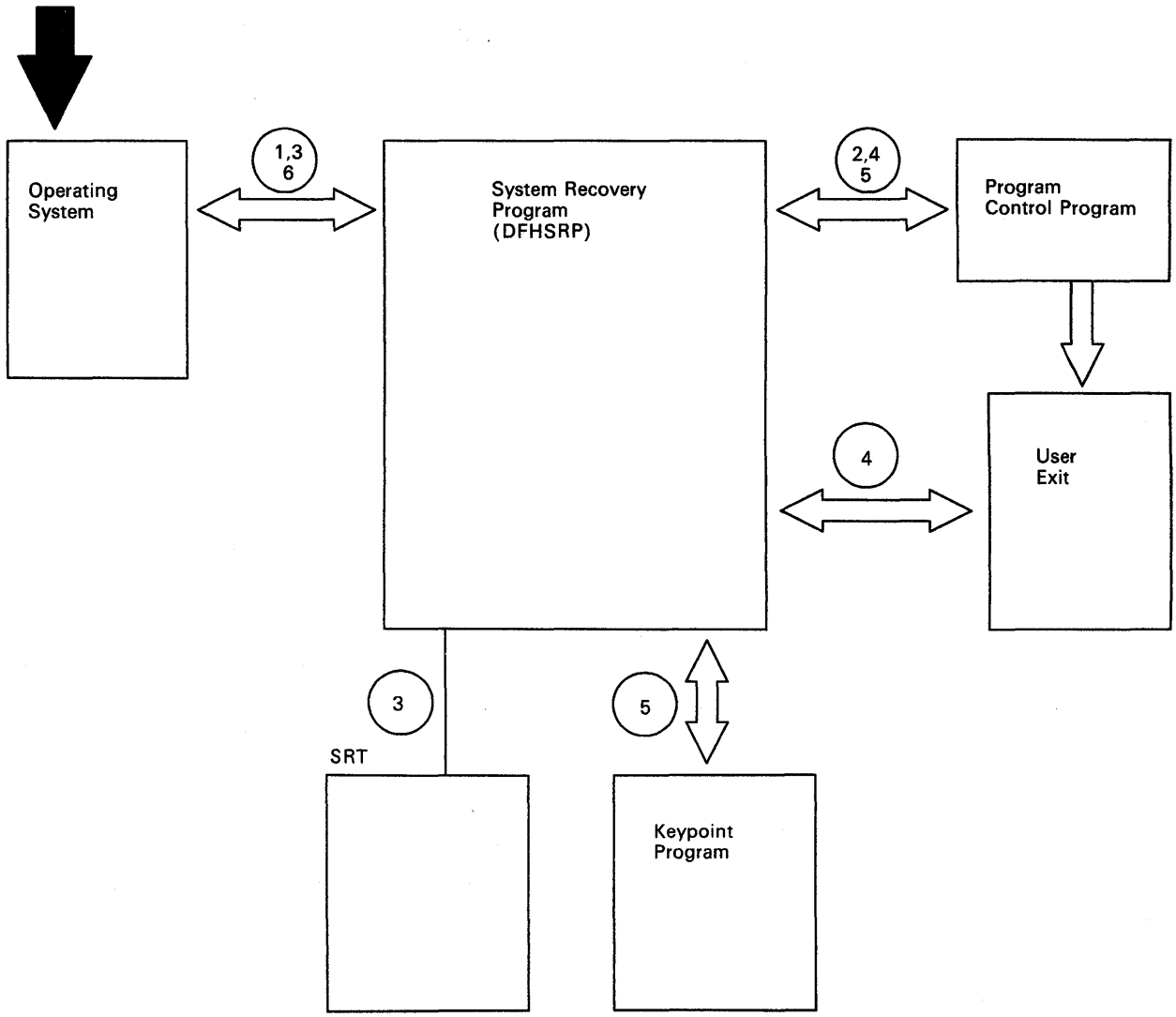


Figure 32. System Recovery Interfaces

Notes:

1. *The system recovery program is given control as an operating system exit routine.*
2. *If the system recovery program determines that a program check or abend condition has occurred during an error condition, or that the current task is a system task, it issues a message, forces a dump, then terminates CICS. Otherwise, it returns control to DFHPCP, via the operating system, to terminate one task with abend code ASRA.*
3. *Invocation of the system recovery program as an ESTAE exit causes it to search the system recovery table for an entry containing the system code.*
4. *If a match is found, the routine specified is invoked or the named program linked to. The routine may be that supplied by IBM as part of DFHSRT.*
5. *Following the recovery logic, a DFHPC TYPE=ABEND macro instruction is issued to abnormally terminate the offending task. If recovery has been achieved successfully, execution of CICS can continue. If a recovery cannot be effected, or if no recovery was attempted, a keypoint is taken (optionally) and CICS is terminated.*
6. *With the high performance option (HPO), the service request block (SRB) in the system queue area (SQA) is freed by using a CICS SVC (DFHCSVC).*

Dynamic Backout Program

The dynamic 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:

- Interval control, 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.
- Task-related user exit recovery DWEs have the DWEDYNB bit set, but the DWEs are not processed.

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:

- For chain records, the program gets the previous record in the log.
- For a sync point record, the program repositions to the last item of **durable data**; this occurs for restartable tasks only. (Durable data refers to data that remains intact after a sync point.)
- For a first input message record, the program passes the record to a user exit. (In addition, if the task is designated as restartable, the program reinstates the record in a TIOA.)
- For TCTUA records and for high-level language communications area records, the program reinstates the records in their original areas; this action occurs for restartable tasks only.
- For a file backout record, GET UPDATE and PUT NEW file management operations are logged. For GET UPDATE a copy of the record before the operation is logged; for PUT NEW the identification of the added record is logged. The dynamic 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.
- For 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 backout program finishes either by transferring control to the abnormal condition program, or by invoking the retry program DFHRTY and then transferring control to DFHACP or invoking DFHPC TYPE=RETRY.

Rollback Function

A user's request for sync point rollback invokes DFHDBP which backs out resources (except for **durable data**) as for dynamic backout.

Figure 33 shows the relationships between the components of dynamic backout.

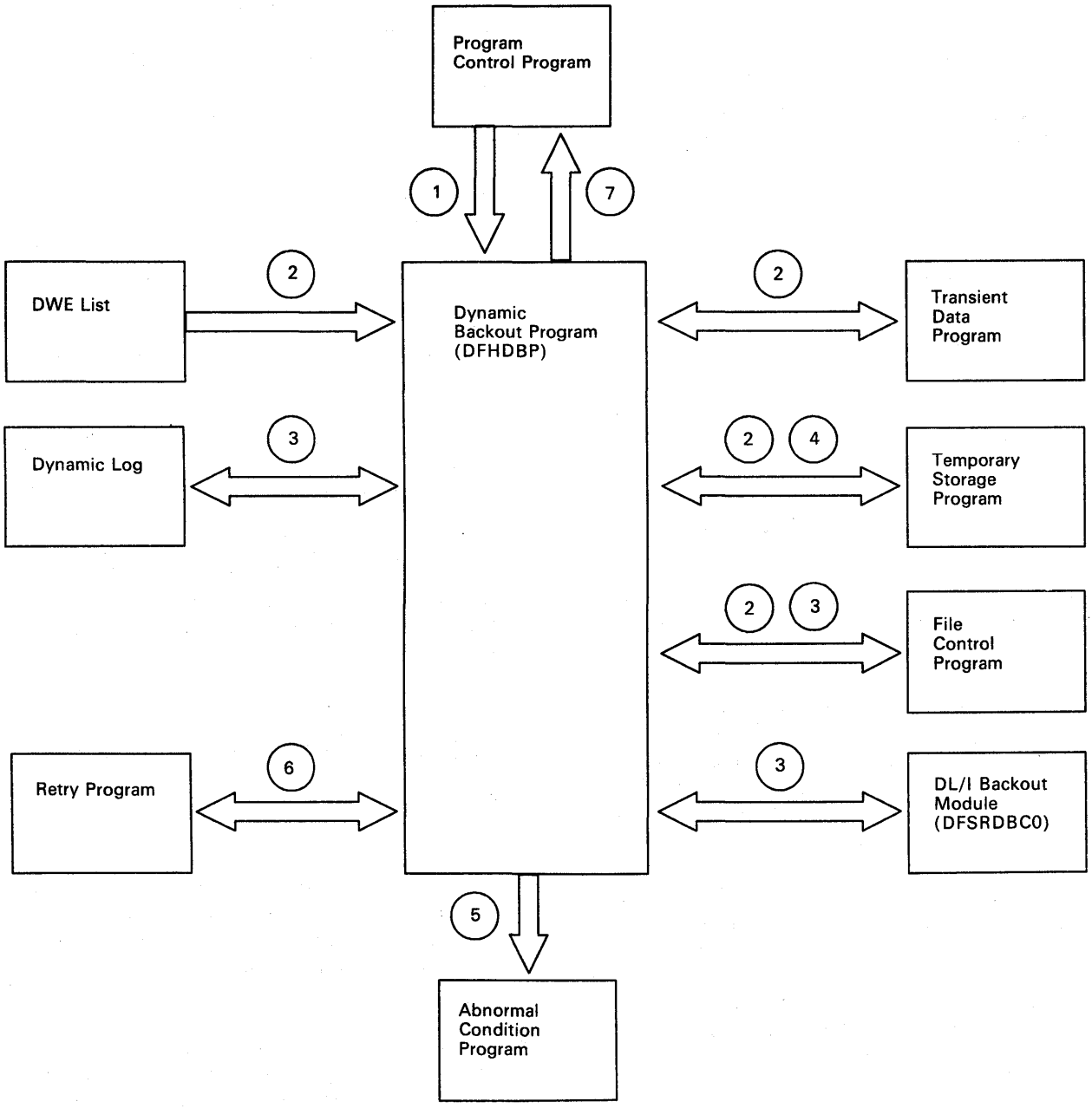


Figure 33. Dynamic Backout Program

Notes:

1. *DFHPCP invokes DFHDBP using XCTL.*
2. *The DWE list is scanned. DFHTDP is invoked for transient data DWEs, and DFHTSP is invoked for temporary storage DWEs, to backout changes to the destinations. DFHFCP is invoked for file control DWEs to perform any outstanding RELEASE.*
3. *The dynamic log is scanned and DFHFC macro instructions issued to reverse file control changes. The DL/I backout module is invoked to backout DL/I changes.*
4. *Spilled dynamic log entries are recovered from temporary storage.*
5. *Transfer control to DFHACP.*
6. *The retry program (DFHRTY) is invoked to decide whether restart is appropriate.*
7. *DFHPC TYPE=RETRY is invoked to restart the transaction.*

Retry Program (DFHRTY)

In the creation of the program control table (PCT), the system programmer can designate selected transactions as **restartable**.

During the execution of any transaction, certain temporary storage data, intrapartition destinations, files, and all DL/I data bases are protected for dynamic backout (described earlier). In addition, for a restartable transaction, the following actions take place:

- Any TIOA, command-level communications area, or terminal user area existing at task initiation is copied to the dynamic log.
- Interval control AIDs used in the task are preserved by means of deferred work elements (DWEs) until the next sync point.
- Flags in the TCA are maintained to show:

  What terminal traffic has occurred during the task.

  Whether a sync point has been passed.

  Whether or not the current activation of the task is the result of a restart.

In the event of an abend and after backout has been successfully completed, the user's DFHRTY exit program is invoked to decide whether the task is to be restarted. (In the absence of DFHRTY, the restart is allowed to proceed only if the abend code is that of a program isolation deadlock (ADLD), no sync points have been taken, and no terminal traffic has occurred.)

If the task is to be restarted, the proceed flag in the TCA is turned on. DFHPCP is then invoked (1) to free all storage areas that can be freed; (2) to increase the retry counter in the PCT; (3) take a sync point; and (4) invoke the initial program for the transaction (as if for an XCTL macro).

If the task is **not** to be restarted, DFHACP issues a message saying why the restart has been rejected and terminates the task.

The sample retry program (DFHRTY) can optionally be rewritten by the user to perform logical tests that will decide whether or not a task is to be restarted. The program has the task's TCA and all user storage available, and can use CICS facilities such as file control and transient data. As a result of its tests, DFHRTY can change the setting of the proceed flag to require or to prevent a restart attempt.

Abnormal Condition Program (DFHACP)

The abnormal condition program (DFHACP) is a system service program used to analyze abnormal conditions that occur within the system and inform the terminal operator of problems. Errors can be classified as belonging in either of two broad categories:

- Task abnormal conditions, which are detected by CICS 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, if possible, informed of the error, and the error is logged at destination CSMT.

- Operator errors, such as invalid transaction identifications, security key violations, or failure of an operator to sign on the system before attempting to communicate with CICS. When this happens, the terminal operator is notified, and the error is logged at destination CSMT.

Figure 34 shows the interfaces between the abnormal condition program (ACP) and other components when an error has been detected.

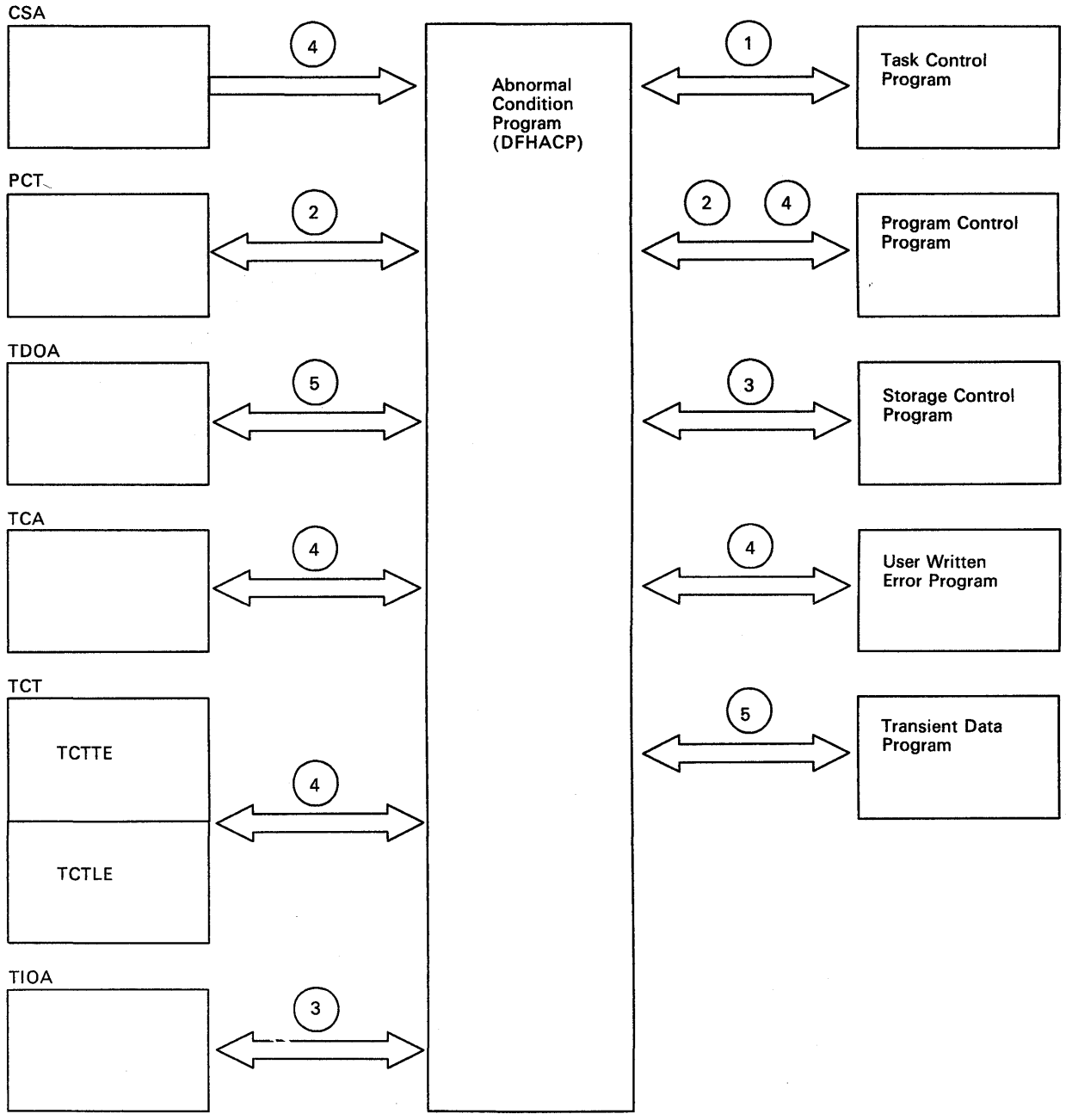


Figure 34. Abnormal Condition Program Interfaces

Notes:

1. *DFHACP is invoked by task control whenever an invalid transaction code is detected.*

2. *DFHACP is invoked by program control whenever a task is abnormally terminated. The operator ID for error messages is in the TCTTE at TCTTEOI. DFHACP returns to program control 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.*

3. *DFHACP communicates with storage control to obtain and release terminal input/output areas (TIOAs).*

4. *If a user-written program error program (PEP) is provided, DFHACP issues a DFHPC TYPE=LINK macro instruction to transfer control to the program so that it can execute its function. All user-written PEPs communicate their requests through the TCA. The CSA is used to obtain the addresses of programs, and the terminal control table terminal entry and line entry (TCTTE and TCTLE) are used to determine the status of the terminal and the line. Any abend within a PEP results in CICS termination.*

5. *Error messages are written to the transient data destination, CSMT, using the message program DFHMGP.*

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 (DFHACP) whenever a task terminates abnormally. The only exception to this procedure is when CICS deliberately terminates a task to alleviate a stall.

The user can perform any kind of corrective action within PEP. CICS 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. For a description of how to write a program error program, see the *CICS/MVS Customization Guide.*

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 also provides a sample DFHTEP.

The user can perform any kind of corrective action within DFHTEP; CICS provides options that either place the terminal out of service or retry the I/O operation. For a description of how to write a terminal error program, see the *CICS/MVS Customization Guide.*

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 (DFHZNEP) for VTAM-supported terminals. CICS provides a sample node error program, which assists the user in the following ways:

- It provides a general environment within which it is easy for users to add their own error processors.
- It provides 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.
- It serves as the default NEP where the user selects a NEP at system initialization.

Details of register usage and relevant DSECTs for user node error programs can be found in the section "User-Written Node Error Programs" in the *CICS/MVS Customization Guide.*

Emergency Restart

CICS provides an emergency restart facility when system execution is interrupted before controlled shutdown can be performed. This 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 of the volume descriptors for standard-labeled tapes
- Recovery and resynchronization of the state with respect to other systems which CICS was communicating with through ISC, or task-related user exits.

Recovery Control Program (DFHRCP)

To recover various resources, each resource manager needs only records from the system log, and, for backout, only those recorded by tasks that were in-flight at the time of CICS failure. For the **forward recovery** of tables in main memory, it needs only those records written by transactions that completed execution.

The recovery control program (DFHRCP) presents these system-log records to the recovery components of resource managers on demand (using the DFHRC macro). To do this, the recovery control recovery program (DFHRCRP) invokes the recovery utility program (DFHRUP). DFHRUP scans the log backward, and copies the relevant log records to the recovery file, using the DFHRC TYPE=WRITE macro. The recovery file is part of the restart data set. It is separated from the other part, the catalog, by the key-range. The records are then extracted from the recovery file by DFHRCP using keyed access.

Figure 35 on page 142 shows the recovery utility program interfaces.

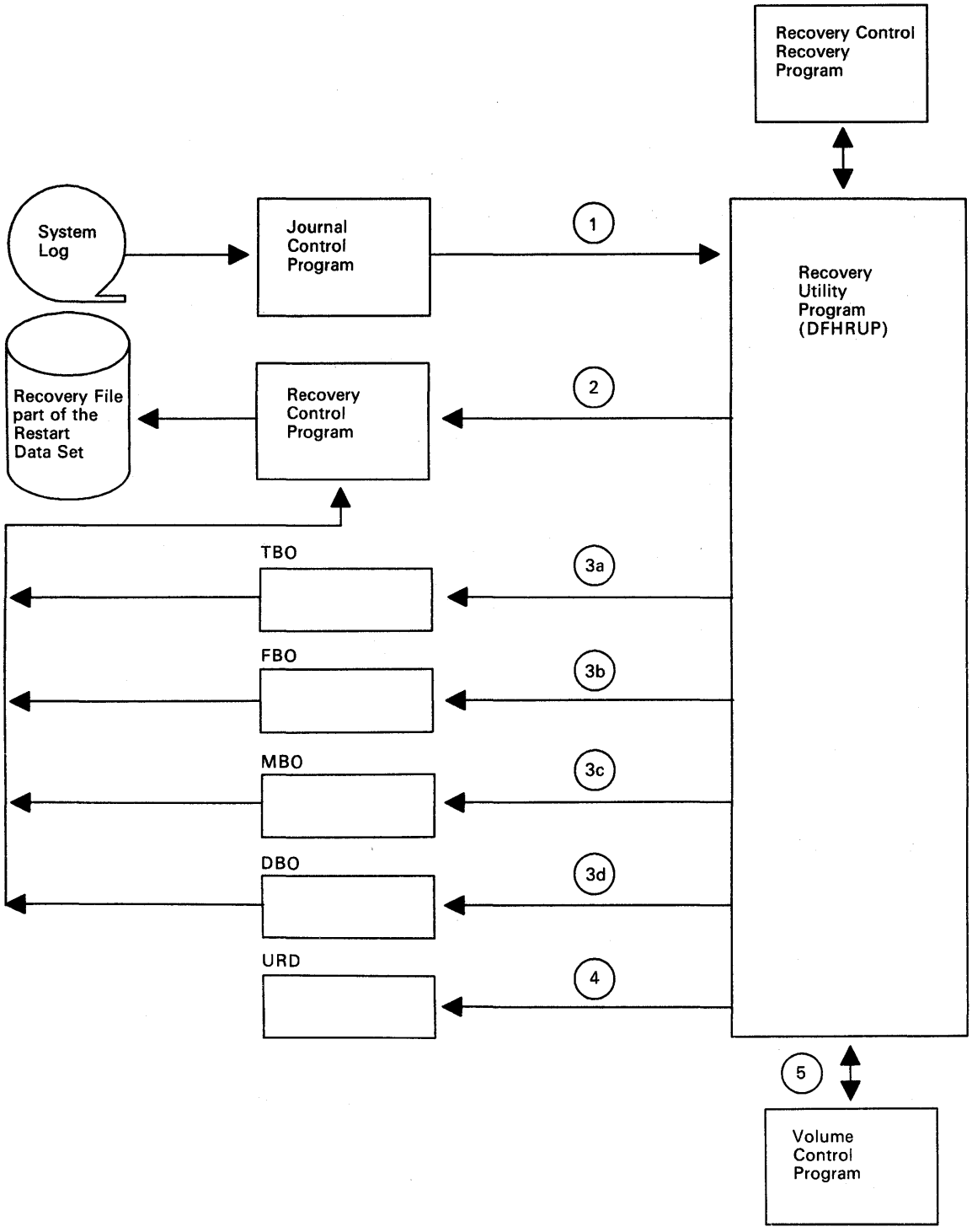


Figure 35. Recovery Utility Program Interfaces

Notes:

1. *During an emergency restart, the system log is automatically repositioned after the last record written during the previous execution. DFHRUP interfaces with DFHJCP to read this data set backward in order to process system recovery data and to collect user recovery backout data. The backward scan is completed when the following conditions are met:*

   a. *At least one complete activity keypoint (delimited by end and start of keypoint records) has been retrieved.*

   b. *The start-of-task record for each logical unit of work (LUW) in-flight at system termination has been reached.*

   c. *All messages that have not received a positive response have been found.*

2. *During the backward scan, DFHRUP uses DFHRCP to output the following data to the recovery file:*

   a. *Records output to the system log by tasks (LUWs) that did not complete processing before the system abnormally terminated (in-flight tasks). These records follow the standard journal control record layout, they have the flag JCSPRRIF set* **on** *in field JCSPP1, and they are as follows:*

      1) *Records automatically logged by the file control program for data sets with the specification LOG = YES in the FCT.*

      2) *Records automatically journaled to the system log by the file control program (FCP), according to the user-specified option in the FCT.*

      3) *User-journaled records written to the system log that were output by in-flight tasks.*

         Note: 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). If the task was completed, the flag JCSPRRIF is* **off** *in field JCSPF1. User-written activity keypoint records should have an identification as stated above in order to be accessible from the restart data set.*

   b. *Initial input and final output message per LUW logged by terminal control program for terminals with the PROTECT option group specified in the PCT.*

   c. *All input/output messages for in-flight tasks journaled by the journal control program as specified in the MSGJRNL= operand in the PCT.*

   d. *All DL/I records logged to the system log that did not complete processing before the system abnormally terminated.*

   e. *All update/replace records entered on the system log by the temporary storage control program, as specified in the temporary storage table (TST).*

   f. *Records output using the DFHRC TYPE=LOG macro by various resource managers for table recovery. They are flagged for* **forward recovery** *in the log. DFHRUP copies them to the recovery file only if the transaction completed.*

3. *The following tables are created by DFHRUP, and later written to the recovery file using DFHRCP:*

   a. *The transaction backout table (TBO) contains an entry for each task in-flight at the time the system abnormally terminated.*

      There are two types of entries in the transaction backout control Record:

      1) *In-flight tasks. These are tasks that caused records to be written to the system log, but failed to complete before system failure. No special start-of-task record is written to the system log, but the first record logged for the task is flagged as being start-of-task. When DFHRUP reads the log backward, and the first record found for a task is one* **other than** *an end-of-task record, this task is considered in-flight. DFHRUP*

must then find the corresponding start-of-task indication to complete the collection of recovery backout data for this task.

2) *Active tasks. These are tasks that completed a LUW and started another, but did not cause any records to be written to the system log during this LUW. Thus, during DFHRUP processing, a completion of a LUW was found, but no* **physical** *end-of-task (that is, task DETACH) was found.*

b. *The file backout table (FBO) contains an entry for each data set for which a logged or journaled record was written to the restart data set.*

c. *The message backout table (MBO) contains an entry for each terminal for which logged or journaled message or message resynchronization records were written to the restart data set.*

d. *The DL/I backout table (DBO) contains an entry for each in-flight transaction that was scheduled to alter a DL/I data base.*

4. *DFHRUP re-creates from the log those unit of recovery descriptors (URDs) that existed before the breakdown. URDs seen in the first complete keypoint encountered during the scan of the log are restored unless URDs with the same resource manager name and recovery token have been obtained from another source. For certain log records, new URDs are created.*

5. *DFHRUP passes all volume data to the volume control program.*

Transaction Backout Programs (DFHFCBP, DFHDLBP, DFHTSBP, DFHTCBP, DFHUSBP)

During emergency restart, to maintain integrity, file control, DL/I, temporary storage, and, under user control, user storage, require the backout of changes made by transactions that did not complete execution. Terminal control requires the recovery of messages. The modules that perform this backout are DFHFCBP, DFHDLBP, DFHTSBP, DFHTCBP, and DFHUSBP. They are invoked by the corresponding resource recovery managers, DFHFCRP, DFHDLRP, DFHTSRP, DFHTCRP, and DFHUSRP. Backout is not required for transient data; the DCT loses uncommitted changes. The same applies to temporary storage, except for temporary storage REPLACEs, which do need backing out.

The backout programs read their own resource backout records, and the corresponding backout tables (FBO for DFHFCBP, DBO for DFHDLBP, MBO for DFHTCBP, TBO for DFHUSBP, and nothing for DFHTSBP) from the recovery file, and apply them to the resource, driving user exits as required.

Keypoint Programs (DFHAKP, DFHWKP)

Figure 36 shows the relationships between the keypoint programs and other components during activity keypointing.

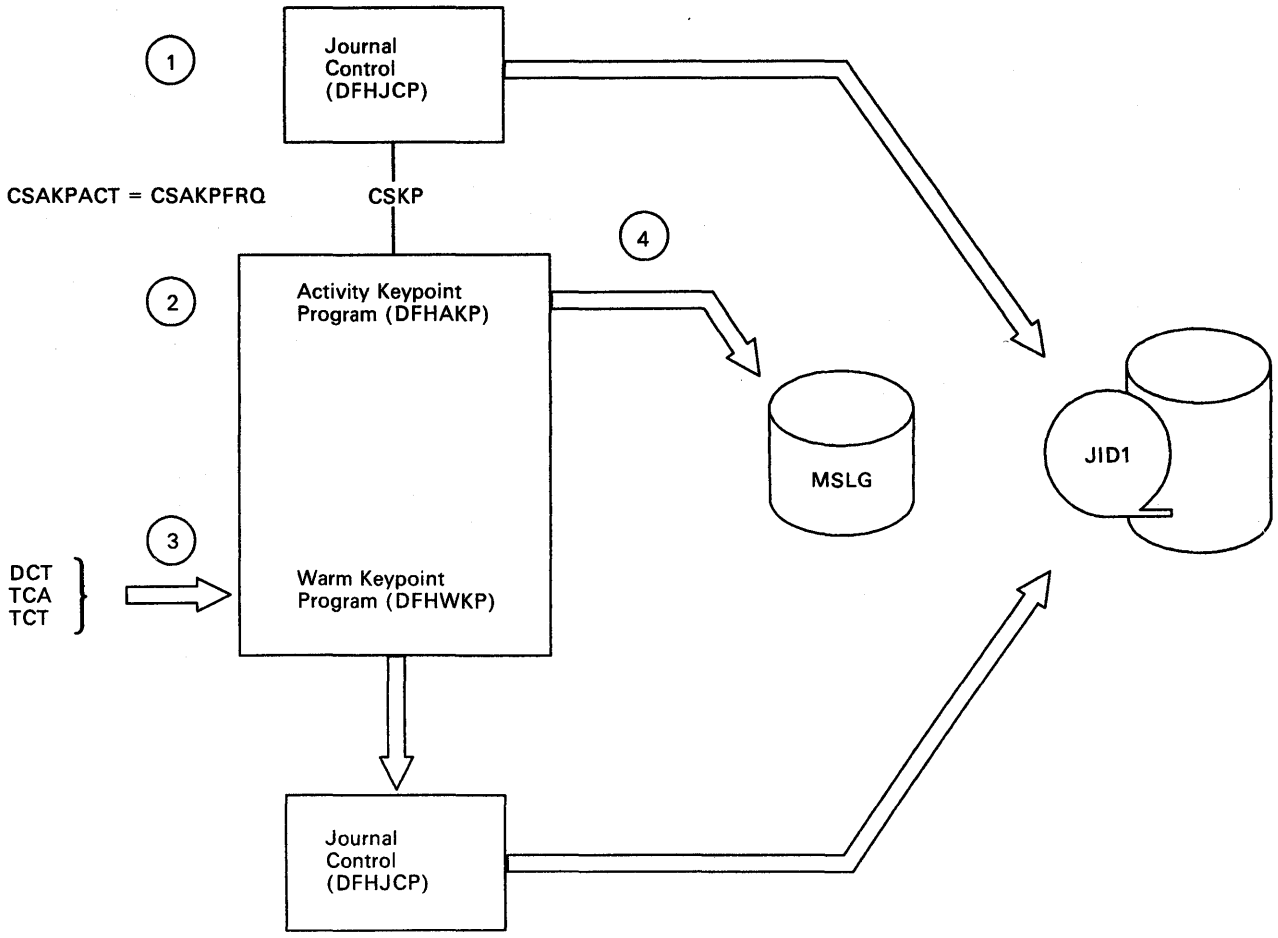


Figure 36. Keypoint Program Interfaces during Activity Keypointing

Activity Keypointing

Notes:

1. *The need for activity keypointing is signaled when an activity keypoint frequency count is reached during journal control (DFHJCP) logging of activity on the system log (JID1) or when the log switches volumes.*
2. *Journal control attaches the task associated with transaction identification CSKP. CSKP invokes the activity keypoint program (DFHAKP).*
3. *DFHAKP gathers the TCA, DCT, and TCT information in buffers, and interfaces with DFHJCP to log this information on the system log.*
4. *DFHAKP writes a time stamp to the master terminal (CSMT).*

Warm Keypointing

The warm keypoint program writes system environment information to the restart data set at normal termination of CICS, so that CICS can be warm-started later.

Task-Related User Exit Recovery

During the execution of application programs, the CICS sync point program communicates with the resource manager task-related user exit to **prepare-to-commit**, to **commit-unconditionally**, or to **backout**. The purpose of these calls is to ensure that changes to recoverable resources performed in a logical unit of work (LUW) are either all committed or all backed out, when there is a failure anywhere in the systems.

Each LUW has a unique name, known as the recovery token, which identifies the CICS system and any other system that communicates with the CICS system. The recovery token is the 8-byte value stored in the TCA at TCARTKN at the start of each unit of recovery. The last byte of the recovery token is set to zero. The form of the recovery token makes it easy for CICS to decide whether it originated before or after a cold start.

When the resource manager receives the **commit-unconditionally** or **backout** call, it takes the corresponding irreversible step, if possible. If the action is successful, the resource manager sends the appropriate return code. If not, it sends a return code which requests that CICS remembers this, and tries to resolve the status at a later time.

CICS remembers the state of the LUW by creating a unit recovery descriptor (URD) for each resource manager that was unable to complete a request. The URDs are chained from the CSA; the start of the chain is CSAURDA, which points to the last URD created.

LUC URDs are logged by the sync point program, and other URDs are logged by the keypoint program. DFHRUP creates URDs from the log tape during emergency restart, if necessary.

A URD exists until the resource manager indicates that it has honored a commit or backout request delivered by CICS following a RESYNC command.

Chapter 2.5. System Support Component

The system support component consists of several functions that are required in order to run CICS. Most of the support functions are performed offline.

- System initialization
- System termination
- Program preparation utilities
- System log/journal utilities
- CSD utility program.

System Initialization Program (DFHSIP)

The system initialization program (DFHSIP) is a non-real-time component of CICS and is resident only long enough to start up CICS.

System initialization provides these options for restarting:

COLD Complete reinitialization of CICS and system data sets. All previous systems activity is ignored, except for existing lists of standard-labeled tapes. The existing lists can also be ignored.

AUTO When the AUTO option is specified, CICS decides what kind of restart is made. If the conditions for a warm restart are all satisfied, then that kind of start is executed. If the restart data set is newly initialized, a cold restart is executed. Otherwise emergency restart is executed.

A warm restart process reinitializes CICS 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 prior to termination, and that a warm keypoint was taken during that termination of CICS.

EMERGENCY This optional restart process restores the system using information recorded during the previous execution of the system to a predefined point which existed prior to the interruption.

STANDBY This start is used by the alternate system in an XRF environment. The alternate system is initialized only to the point where it can monitor the active system. It has surveillance and tracking mechanisms, which read the active system's surveillance signals from the control data set, and information about terminals in the system from the message data set. The alternate system is not fully initialized because, when it takes over from the active system, it needs resources that can only be used by one system at a time, and these are being used by the active system.

Selection of startup options can be made in the system initialization table specification (COLD, AUTO, or STANDBY), or in the override parameters (COLD, AUTO, or EMER).

System Initialization Modules

The main initialization program is DFHSIP. DFHSIP calls a series of overlays DFHSIA1, DFHSIB1, ... DFHSIJ1, which complete initialization. DFHSIP receives control from the operating system, scans the override parameters for CSA and SIT parameters, and loads the appropriate CSA and SIT.

DFHSIA1

DFHSIA1 processes the override parameters.

DFHSIB1

DFHSIB1 loads the CICS nucleus modules, initializes the CSA, and builds the trace table. DFHTEOF may be called during emergency restart to find the end of a system log tape that has a damaged end. If VTAM = YES, DFHSIB1 builds a receive-any RPL pool (RACE).

DFHSIC1

DFHSIC1 initializes the user exit table, opens the restart data set, and determines what type of start is required. If XRF = YES, DFHSIC1 signs on to the CICS availability manager (CAVM).

DFHSID1

DFHSID1 allocates transient data control blocks.

DFHSIE1

DFHSIE1 calculates and builds the VSAM LSR pool if DL/I is used.

DFHSIF1

DFHSIF1 initializes terminal control by opening BTAM data sets, opening the VTAM ACB, and performing a warm start for the TCT.

DFHSIG1

DFHSIG1 opens the dump data set.

DFHSIH1

DFHSIH1 attaches the journal control subtask, builds the page allocation map, calculates OSCOR, calls DFHDLQ to initialize DL/I, and allocates temporary storage control blocks.

DFHSII1

DFHSII1 establishes exits in DFHSRP, performs warm start functions for temporary storage, ICEs, AIDs, and the CSA, attaches the DFHVSP task if required, initializes the table management program, opens journal data sets, initializes system console support, and initializes auxiliary trace. DFHSII1 attaches the III task, and passes control to the terminal control program DFHZDSP.

DGHSIJ1

DFHSIJ1 executes the programs in the program list table (PLT), calls DFHVCP for standard-labeled tapes, calls DFHDLX to complete DL/I initialization, and returns using program control.

The III task

If XRF = NO, the III task reopens the restart data set (RSD), attaches the resource manager tasks, waits for recovery to complete, processes resident programs, and passes control to DFHSIJ1.

If XRF = YES, and START ¬ = STANDBY, the surveillance task (DFHXRSP) is attached.

If XRF = YES, and START = STANDBY, the surveillance task (DFHXRSP) and the console communication task (DFHRCSP) are attached. DFHSIJ1 attaches the transient data and temporary storage resource manager tasks, waits for takeover

completion, opens the restart data set, and attaches the remaining resource manager tasks.

System Initialization Overlays

User-written overlays can be added to the system initialization program; however, they must conform to CICS naming conventions. All system initialization overlays are 7-character names in the format, DFHSIxy, where x is a letter from A to Z and y is a number from 1 to 9. IBM reserves suffixes that end in 1 (A1, B1,...Z1). User overlays may use any other 2-character suffix.

Overlay processing in system initialization is driven from the system initialization table SIMODS = parameter. User overlays can be inserted at any point in system initialization, but the sequence of CICS overlays must not be disturbed. CICS is responsible for common subroutine and overlay linkage (assuming that these routines are not modified), and normal system initialization functions.

A list of system initialization subroutines, and the conventions for calling them, is in the *CICS/MVS Customization Guide*. The following areas are always addressable to system initialization overlays at entry and must be addressable at exit:

Register	Area
13	Common system area (CSA)
5 (SIPPLAR)	System initialization table (SIT)
4 (SIPBAR1)	System initialization common area (SIPCOM)
	System initialization common routines

In addition, at entry to an overlay, registers 3 (SIPBAR2) and 9 (SIPBAR3) contain the entry point address of the overlay and that address plus 4096, respectively, for addressability purposes.

The following fullword fields are supplied as parameter-passing fields between user overlays of system initialization. These fields are not used by CICS overlays:

SIPARMP6
SIPARMP7

Figure 37 shows the relationships between the components of system initialization.

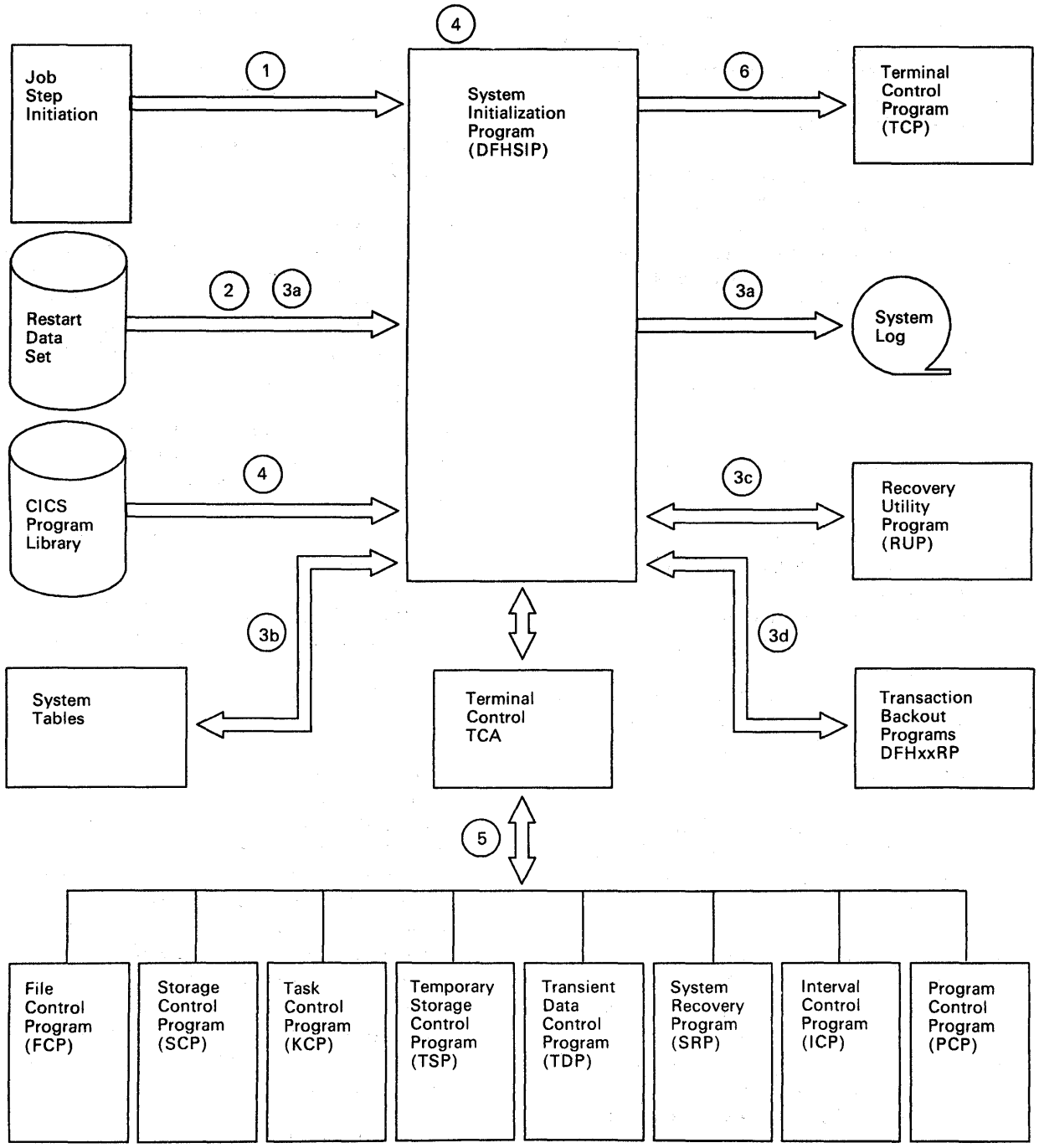


Figure 37. System Initialization Interfaces

Notes:

1. *System initialization receives control from the operating system. Parameters may be passed to system initialization through the PARM operand of the EXEC statement by which it is invoked. (These parameters are documented in the CICS/MVS Operations Guide.)*

2. *The restart data set is used by the warm keypoint program (DFHWKP) 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 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 byte addresses for temporary storage auxiliary destinations/queues*
   - *Temporary storage space allocation bit map*
   - *Interval control elements (ICE) and automatic initiate descriptors (AID)*
   - *Volume descriptors.*

3. *When emergency restart is invoked via the START=EMER or START=AUTO keyword, the system initialization program takes the following actions:*

   a. *Repositions the system log.*
   b. *COLD starts the PPT, PCT, TCT, FCT, DCT, CSA, transient data bit map and the temporary storage maps.*
   c. *Links to the recovery utility program which reads the system log and builds recovery data and tables which are written to the restart data set.*
   d. *Links to the transaction backout programs (DFHxxRP) which read the recovery data and back out the effects of transactions in progress prior to system termination.*
   e. *Reads the status of each journal from the catalog in the restart data set, and reestablishes it in the system, unless JSTATUS=RESET was specified in the initialization override.*

4. *The system initialization program does the following:*

   a. *Builds the CICS nucleus. For the high performance option (HPO) certain modules are loaded in protected storage and control blocks required for service request block (SRB) processing are constructed in the system queue area (SQA). Authorization is relinquished after this has been completed.*
   b. *Initializes data sets.*
   c. *Opens system and user data sets.*
   d. *Constructs and initializes tables.*
   e. *Builds the CICS dynamic storage pool.*
   f. *Accesses the CICS program library by means of BSAM READs to build the CICS nucleus, load tables, load resident application programs, and initialize the processing program table (PPT).*
   g. *Creates indexes for transactions, profiles, programs, map sets, and partition sets.*
   h. *Adds the sets of table entries for the CICS definition file.*

5. *Interface to CICS nucleus modules is required during postinitialization processing, both by system initialization and by application programs running at this time. System initialization always interfaces with storage control, task control, interval control, and program control and may interface with temporary storage control, transient data control, file control, and system recovery. All interface to CICS nucleus modules is done under terminal control's task control area (TCA), which is borrowed temporarily as a communication vehicle.*

6. *System initialization passes control to the terminal control program.*

Task Structure for Initialization

Figure 38 illustrates the task structure during the latter stages of initialization, when XRF = NO was coded in the SIT.

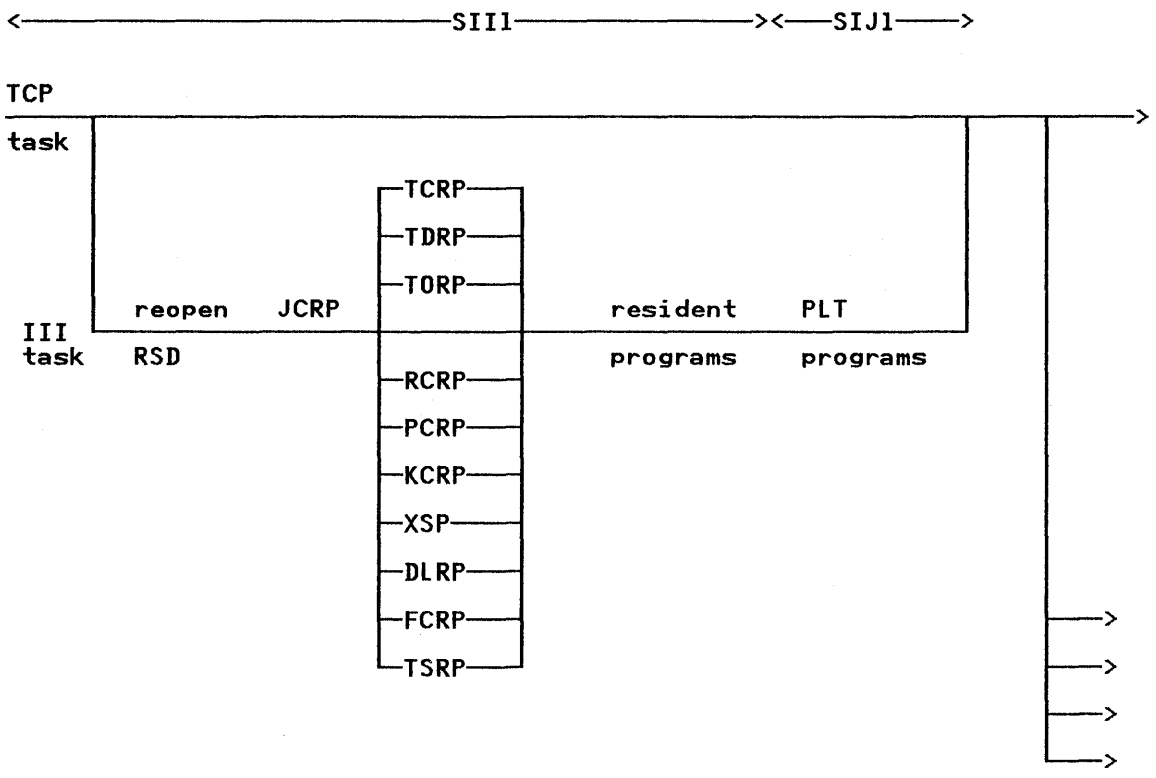


Figure 38. Initialization structure – XRF = NO in the SIT

You see that DFHSII1 attaches a system task, called the III task, which performs recovery and customization.

The TCP task bypasses the BTAM scans of the TCT until control is given to CICS. It invokes the VTAM scan, but has nothing to do.

Active system

Figure 39 illustrates the task structure during the latter stages of initialization of an active system.

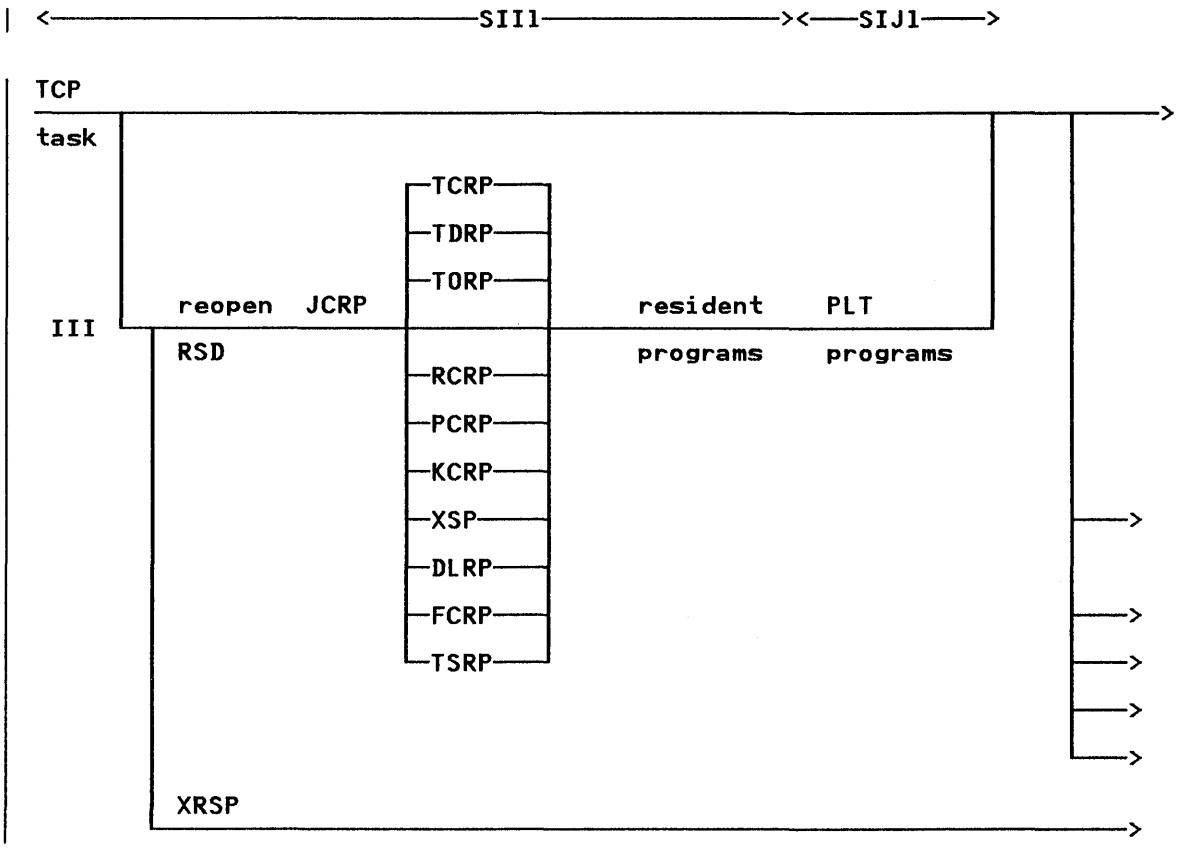


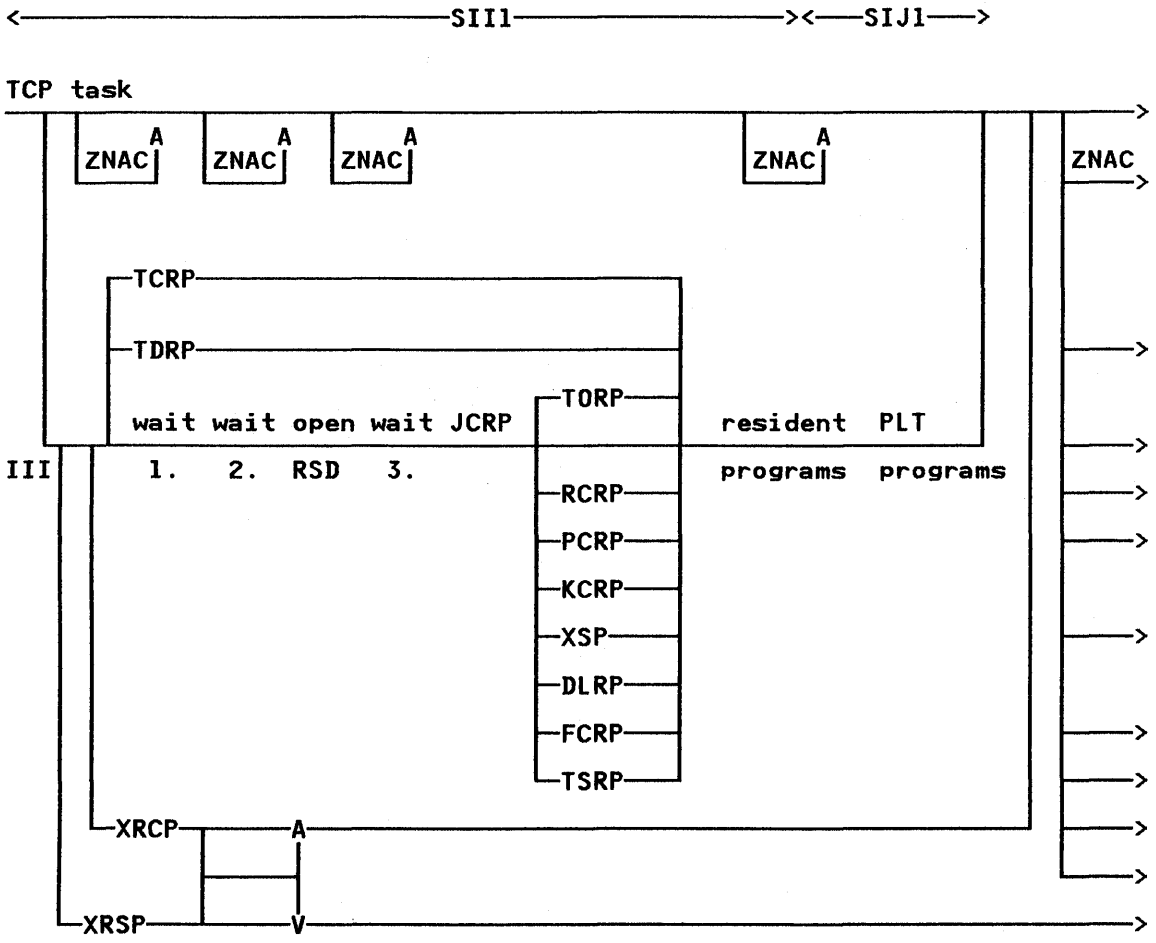
Figure 39. Initialization structure – active system

The picture is very similar to the previous figure.

The only difference is that the XRF surveillance task (DFHXRSP) is attached, because XRF = YES was specified in the SIT. This task runs throughout CICS execution and is responsible for reacting to the events notified to it by the CAVM.

Alternate system

Figure 40 illustrates the CICS task structure during the latter stages of initialization of an alternate system.



Notes:

1. *The first wait is for takeover initiated.*
2. *The second wait is for takeover complete.*
3. *The third wait is for time-of-day clock synchronization.*

Figure 40. Initialization structure – alternate system

This picture is very different from the previous figure.

The XRF surveillance task (DFHXRSP), and the XRF console communication task (DFHXRCP) are attached. The console communication task provides **CEBT** services, and runs throughout the latter stages of initialization. CEBT commands enable the operator to control the alternate system from the console. See the *CICS/MVS XRF Guide* for more details of this command.

The III task attaches the transient data and terminal control recovery tasks. It then waits until takeover occurs.

The transient data recovery task creates the CXRF destination to which early messages to CICS-defined destinations, such as CSMT, are routed. Like the III task, it waits until takeover occurs.

The terminal control recovery task uses CAVM message services to track installations, logons, and logoffs of terminals as they occur in the associated active system.

Processing of standby binds for alternate systems is very similar to that for binds in non-XRF systems. TCTTEs are queued for OPNDST processing by the TCP task. Subsequently, the TCTTEs are queued for ZNAC processing.

Takeover can be initiated from either the console communication task, or the surveillance task. The terminal switch scan task is attached when the takeover request has been accepted by the CAVM. The TCTTEs are queued for further ZNAC processing.

When takeover is complete (that is, when the active system has been canceled), the alternate system (or active system as it is now) can attach the remaining recovery tasks, and complete initialization.

Session clean up runs in parallel with resource recovery and customization. This may involve further ZNAC processing. However, session recovery is deferred until control is given to CICS.

System Termination Program (DFHSTP)

The purpose of system termination (DFHSTP) is to provide for an orderly shutdown of CICS. It is activated by the master terminal program (DFHEMB or DFHMTPA) when that program is responding to a shutdown request entered by the CICS master terminal operator.

Figure 41 on page 158 shows the relationships between the components of system termination.

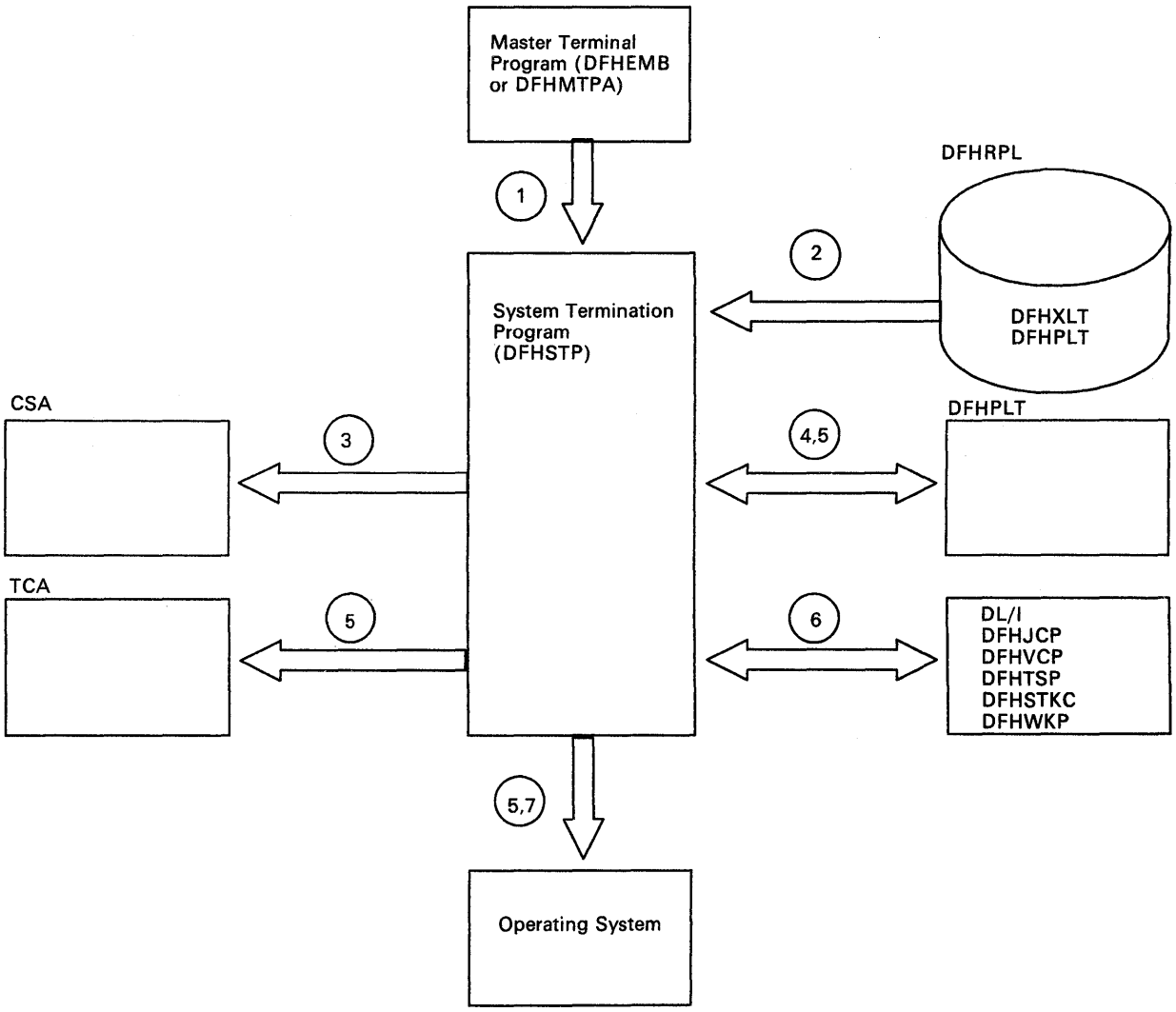


Figure 41. System Termination Interfaces

Notes:

1. *The master terminal program (DFHEMB or DFHMTPA) transfers control to DFHSTP by means of a program control DFHPC TYPE=XCTL macro instruction when a CEMT (or CSMT) shutdown request has been entered by the CICS master terminal operator. The IRC session is quiesced.*

2. *The transaction list table (XLT) and program list table (PLT) are loaded via program control from the CICS program library (DFHRPL).*

3. *Terminal activity is quiesced via an indicator in the CSA. This tells terminal control 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.*

4. *The termination task allows all other tasks (except any journal tasks) to complete before linking to the first program specified in the first portion of the PLT.*

5. *When all programs in the first portion of the PLT have executed, terminal activity is quiesced completely, using bit CSATQIM in CSASSI2 in the CSA. If monitoring is running, it is stopped. The ICE and AID chains are broken (addresses saved in the TWA), and the programs specified in the second portion of the PLT are executed.*

6. *CICS-DL/I interface and journal control and volume control (subtasks) are terminated; temporary storage control (DFHTSP) is requested to output its buffer; statistics are taken by means of a link to the system statistics program (DFHSTKC); and a keypoint is taken by the warm keypoint program (DFHWKP).*

7. *Control is returned to the operating system, with or without a dump (depending upon the parameters specified in the shutdown request causing termination).*

   For the high performance option (HPO), the service request block (SRB) in the system queue area (SQA) is freed by using a CICS SVC (DFHCSVC).

   If an immediate shutdown is requested, no load of tables, terminal quiescing, or execution of programs specified in the PLT occurs (see notes 2, 3, 4, and 5).

Program Preparation Utilities

High-Level Language Preprocessor

The high-level language preprocessor program (DFHPRPR) runs offline to process a high-level language program that contains CICS macro instructions into a form suitable for input to the assembler. The assembler then generates high-level language statements from the CICS 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 "ALL". If this scan fails then the source record is written out with a REPRO card preceding it. DL/I calls are changed to CICS calls; CICS macro instructions are left unchanged except for some conversions from hexadecimal to decimal. An END statement is appended to the output.

Command-Language Translator

The command-language translator is a utility program that runs offline to translate CICS application programs that use the command-level CICS requests. It converts the EXEC statements into CALL statements in the language in which the EXEC CICS statements are embedded. Versions of the translator program are available for:

- COBOL (DFHECP)
- PL/I (DFHEPP)
- Assembler (DFHEAP).

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:

```
EXEC CICS function... termination
```

are recognized.

Keywords and options are recognized and the keywords and parameters converted to a string of bits. The translator converts the statement using a simple list of keywords. Each keyword in the table is specified to be either optional or mandatory and a check is made that all mandatory keywords are present. The translator also has lists of pairs of **positive implications**, that is, if one keyword is present, then the other keyword must also be present; and a list of pairs of **negative implications**, that is, keywords that are mutually exclusive.

The replacement string for each EXEC CICS command is built up in another string. In the case of PL/I, the replacement field contains a single CALL statement. In the case of COBOL, the string contains a series of MOVE statements followed by a CALL statement.

Errors in the source can 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 data sets to be used as system logs or journals. They also allow the user to place an end-of-file mark on magnetic tapes holding parts of 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 preformatted 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 the message data set
- Acquires 32K of storage for a record area
- Opens the 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.

Format Disk Utility

The format disk utility (DFHJCJFP) is a stand-alone program which prepares a contiguous disk data set for use in a system log or journal. A preformatted record is written on each track of the data set.

This guarantees that, for all later reads, only journal blocks or formatting records will appear. Consequently, a CICS restart can always locate and reposition to the most recently written record in a disk journal. The repositioning is based on date and time stamps.

Tape End of File

The tape end-of-file program (DFHTEOF) is run as a stand-alone 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 run before system failure
- Writing end of file.

The journal 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 that they belong to the same journal, and the same part of that journal. Verification of these label records is performed as follows:

- Creation date equal to that specified on the volume label.
- Volume sequence number equal to that specified on the volume label.
- Run start time equal to that specified on the volume label.

The end of valid journal 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 for the IBM 3480 tape, the tape is rewound, closed, opened, and read forward, counting to the correct record, before writing a tape mark. If either condition occurs for other tapes, the tape volume is backspaced over the appropriate number of records and a tape mark is written.

For standard-labeled journal tapes, the volume identifier is specified by the calling parameter or by JCL. If DFHTEOF is called by DFHSIP, DFHSIP passes the volume identifier to it.

Journal Print Utility (DFHJUP)

The journal print utility program (DFHJUP) is used with CICS and its related data bases. Its main function is to examine, display, and transfer data from the CICS log. DFHJUP:

- Prints or copies a complete log data set.
- Prints or copies multiple log data sets based on control statement input.
- Selects and prints log records based on the position in the data set.
- Selects and prints log records based on data contained in the record, such as the time, date, or identification field.
- Allows exit routines to process any selected log records.

These facilities are selected and controlled by a series of statements that allow you to define the input and output options, selection ranges, and various field and record selection criteria.

The program has two phases:

1. Control statement processing – record test and selection parameters are constructed, and control statement errors are diagnosed.
2. Record selection and output processing – the input data is read, analyzed, and compared with the selection parameters to determine the applicability of the record output.

During the first phase, control statements are read and examined, and the required test, or series of tests, is constructed to create a test group. This test group is then used in record selection when control passes to the second phase of the program. During the second phase, the input data records are read, and disposition is decided by the results of each test in the group. When the end of the input data is reached, either by an end-of-file condition being detected, or the indicated record count being achieved, control returns to the first phase, where the next group of tests is constructed.

CSD Utility Program (DFHCSDUP)

The CSD utility program (DFHCSDUP) provides services for the CICS system definition file (CSD). The utility command processor (DFHCUCP) validates commands and invokes the appropriate routine to execute the requested function. DFHCSDUP is an offline program and calls DFHDMP to access the CSD.

The commands performed by DFHCSDUP are:

ADD
Add a group to a list.

UPGRADE
Add or change IBM-supplied definitions to the CSD, on release upgrade.

INITIALIZE
Prepare a newly defined data set for use as a CSD file. Standard entries for IBM-supplied source definitions are created on the CSD file. These definitions are arranged into two group lists, DFHLIST and DFHLIST2, which are created containing the names of the basic groups required to bring up and run CICS. A control record is also created at the start of the CSD file which identifies the CICS release, current level of service, and other accounting information.

MIGRATE
Transfer the contents of the PCT, PPT, and TCT tables from a CICS load library to the CSD file.

COPY
Copy the resource definitions for a group to another group. Also, copy definitions from one CSD to another.

APPEND
Add the names of groups in a list to another list. If the receiving list does not exist, it is created. If it does exist, the contents of the first list are added to the end of this list. No duplicate group names are allowed in a list. APPEND also transfers a list from one CSD to another.

DELETE
Erase all the resource definitions in a group, or all the group names in a list. If a list is erased, the groups named in the list are not erased.

LIST
List the current status of CSD files.

VERIFY
Remove internal locks on groups and lists. These locks are normally removed when a command completes, but VERIFY provides a tool for use when a command fails to complete.

SERVICE
Carry out preventative or corrective maintenance to a CSD file. This is performed by loading and running the program DFHCUS1B.

When DFHCSDUP is invoked, control is handled by DFHCUCP. DFHCUCP takes a command from the SYSIN data stream, using DFHCUCB to obtain the command, and DFHCUCA to analyze, and parameterize, it. Some syntax errors are diagnosed, and reported, by DFHCUCA, and further contextual validation takes place in DFHCUCV. Valid commands are then passed to the relevant service program for execution, for example, a MIGRATE command is handled by DFHCUMIG. If command execution is successful, the next command is processed. All commands in the data stream are validated, but the execution of commands stops when an invalid command is encountered. Execution of subsequent commands is also suppressed if an error of severity 8 or higher occurs when the command is executed.

If errors occur while processing commands, they produce error messages in the DFH51xx and DFH52xx series, which are written to the SYSPRINT file. If DFHCSDUP terminates abnormally, an operating system abend occurs, and a dump is produced. DFHCSDUP uses the routines of DFHDMP to process interactions with the CSD. All CSD management functions use DFHDMP05, which performs the environment-dependent VSAM operations on the CSD file. DFHDMP05 also processes all interactions with operating system services.

Chapter 2.6. Application Services Component

CICS provides several functions designed to perform services closely associated with user applications. These services rely on CICS 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
- Built-in functions
- Execution diagnostic facility (EDF)
- Command interpreter.

Basic Mapping Support

The basic mapping support (BMS) function allows the CICS application programmer to have access to input and output data streams without including device-dependent code in the CICS application program.

Maps, map sets, and partition sets are assembled offline using CICS 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 which 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, and so on, 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. The extended attributes can also be modified if maps are defined with EXTATT = YES. 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 an IBM 3270 Information Display System 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 fields from a map that can be keyed or selected to be changed, without making it necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS 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 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 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 terminal of the IBM 3270 Information Display System 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 message-switching transaction.

A BMS map which specifies extended attributes can be used for terminals which do not support extended attributes. When sending data to a variety of terminals, some of the terminals may support extended attributes and others may not. When a BMS ROUTE request is processed, BMS looks at the TCTTEs for all specified terminals and constructs a set of all supported attributes.

A data stream is produced by BMS using this set of attributes, and the data stream and set of attributes for each page are written to a temporary storage record. When the page is later read from temporary storage, the data stream for each terminal is modified, if necessary, to delete attributes not supported by that terminal.

Terminal Paging

This facility 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 output without regard for the record size imposed by the output terminal.

CICS provides the terminal operator with a generalized page retrieval facility that can be used to retrieve and dispose of pages.

Device Independence

This facility allows the user to prepare output without regard for the control characters required for message heading, line separation, and so on. 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 inserts the appropriate leading characters, carriage returns, and idle characters and eliminates trailing blanks from each

line. If the device does not support extended attributes, the extended attributes are ignored.

CICS allows the user to set horizontal and vertical tabs on those devices which support the facility (for example, the IBM 3767 Communication Terminal, and the IBM 3770 Data Communication System). For such devices, CICS 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 (BMS) is provided by means of a number of modules, each of which interfaces with other BMS modules, CICS management components, and application programs. The maps that are handled by BMS may be new maps, created to utilize BMS mapping capabilities. The interrelationships of CICS programs requesting mapping services are summarized in Figure 198 on page 384. Additional details for specific programs within basic mapping support are given on the pages that follow.

There are three versions of BMS selected by the DFHSIT options:

BMS = MINIMUM | STANDARD | FULL

The module used by the minimum version of BMS is:

- Mapping control program (DFHMCP).

The additional modules used by the standard version of BMS are:

- Data stream builder (DFHDSB)
- Non-3270 input mapping (DFHIIP)
- Fast path module (DFHMCX)
- LU1 printer mapping (DFHML1)
- 3270 mapping (DFHM32)
- Page build program (DFHPBP)
- Partition handling program (DFHPHP)
- Route list resolution (DFHRLR)
- Terminal page program (DFHTPP).

The additional modules used by the full version of BMS are:

- Terminal page cleanup (DFHTPQ)
- Terminal page retrieval (DFHTPR)
- Terminal page paging (DFHTPS).

Data Stream Build (DFHDSB, BMS)

The data stream build program addresses the page buffer, which was composed by the page and text build program (DFHPBP). The page buffer contains lines of output data that are to be written to a terminal other than an IBM 3270 Information Display System. The number of lines is contained in the TTPLINES field. 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 format management header (FMH) for some VTAM-supported devices.
- Allows horizontal and/or vertical tab processing.

Figure 42 on page 166 shows the relationships between the components of data stream build.

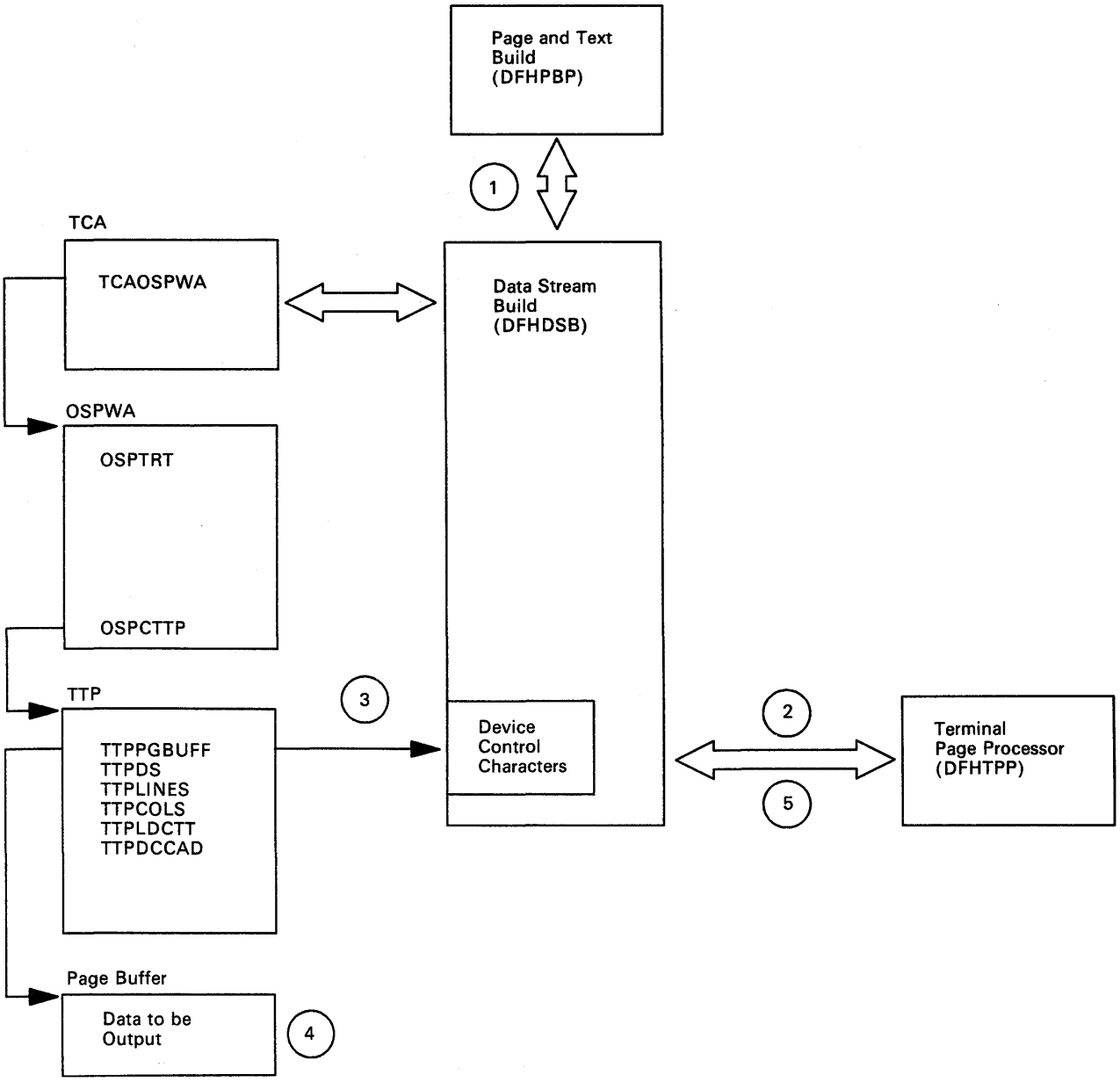


Figure 42. Data Stream Build Interfaces

Notes:

1. *DFHDSB is entered from the page build program to process the page buffer.*
2. *For TYPE=NOEDIT, page buffer compression is skipped and control returns to DFHPBP which calls the terminal page processor (DFHTPP).*
3. *If not TYPE=NOEDIT, the appropriate device control characters for the target device are selected for substitution.*
4. *The page buffer containing the data to be compressed is located through the address stored at TTPPGBUF.*
5. *After compression of the page buffer data, control returns to DFHPBP which calls DFHTPP to provide disposition of the page.*

Non-3270 Input Mapping (DFHIIP, BMS)

The non-3270 input mapping program (DFHIIP) is called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.

Figure 43 on page 168 shows the relationships between the components of non-3270 input mapping.

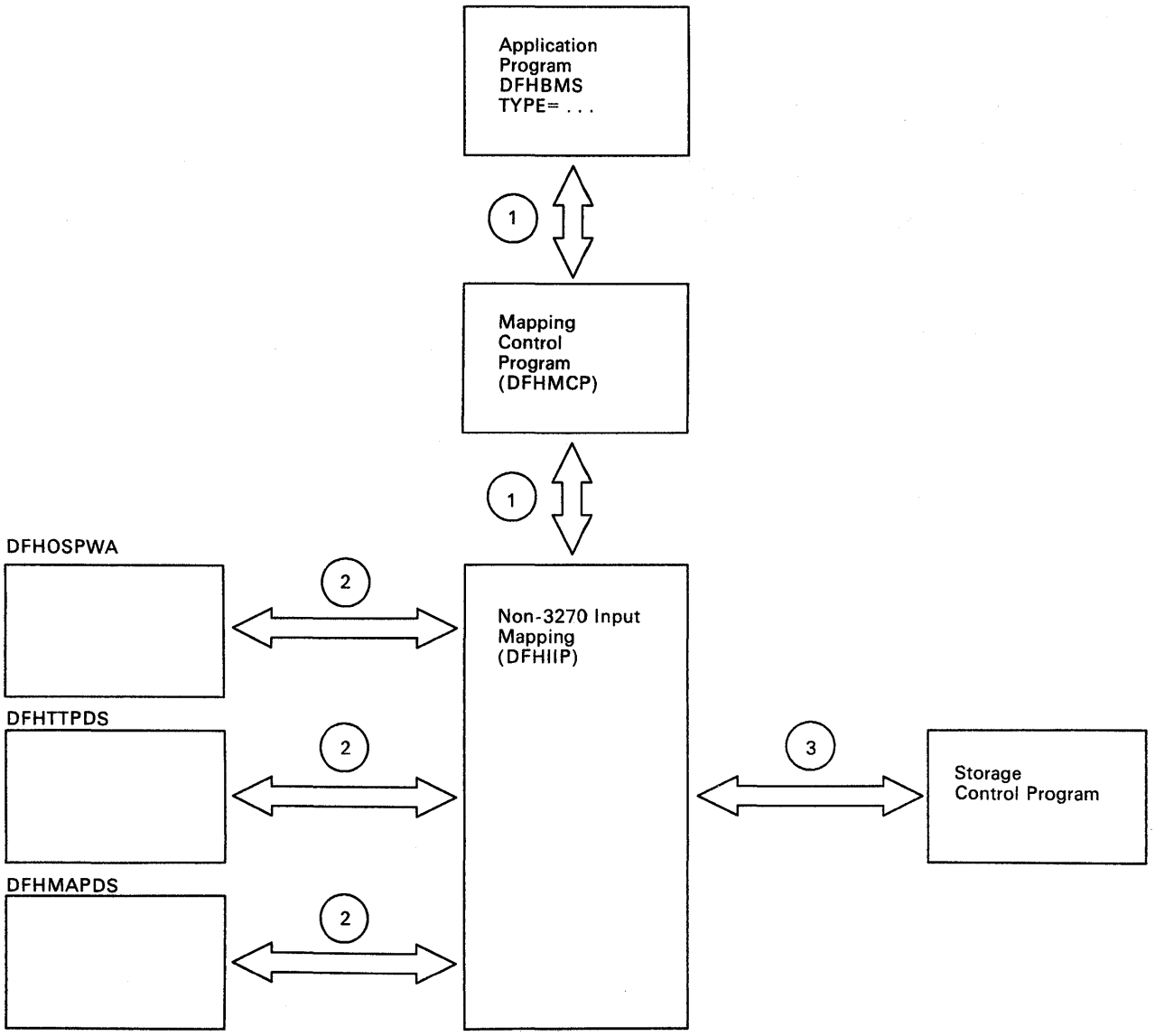


Figure 43. Non-3270 Input Mapping Interfaces

Notes:

1. *A DFHBMS TYPE=IN or TYPE=MAP request by an application program, communicating with other than an IBM 3270 Information Display System, passes information via the TCA through the mapping control program (DFHMCP) to DFHIIP.*
2. *The map required for an operation is either passed by the application program or loaded by DFHMCP.*
3. *DFHIIP communicates with storage control (DFHSCP) to obtain and release buffers for mapping operations.*

Mapping Control Program (DFHMCP, BMS)

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.

Figure 44 on page 170 shows the relationships between the components of mapping control.

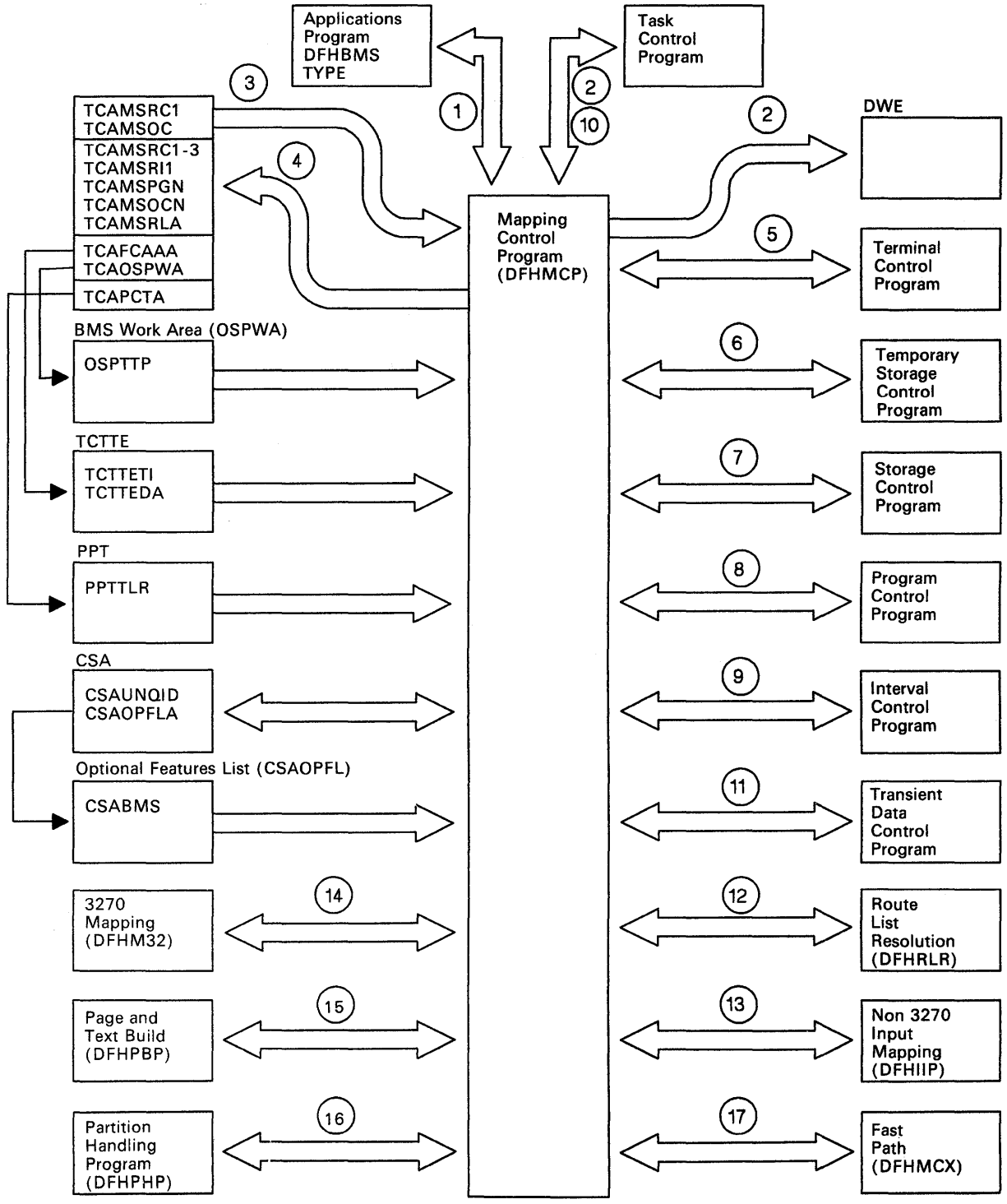


Figure 44. Mapping Control Program Interfaces

Notes:

1. *This program is entered when an application program issues a DFHBMS request for basic mapping support services.*

2. *It may also be called by task control 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.*

3. *The expansion of the DFHBMS macro instruction in the application program inserts data in TCA fields labeled TCAMSxxx.*

4. *The following information is returned to the requestor in fields of the TCA: error codes, page overflow information, and (if TYPE=RETURN was specified in the request), a list of completed pages.*

5. *A terminal control DFHTC TYPE=SAVE macro instruction is issued if TYPE=SAVE was specified in the DFHBMS macro instruction.*

6. *DFHMCP communicates with temporary storage control to put 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.*

7. *DFHMCP communicates with storage control to:*

   a. *Acquire and free storage in which the MCR is built (TYPE=PAGEOUT after TYPE=STORE and/or TYPE=ROUTE).*

   b. *Acquire and free storage in which to copy the message title (TYPE=ROUTE,TITLE=symbolic address or YES).*

   c. *Acquire storage to build automatic initiate descriptors (AIDs) for nonrouted messages or routed messages to be delivered immediately (TYPE=PAGEOUT).*

   d. *Acquire a BMS work area (OSPWA) at the time of the initial BMS request.*

   e. *Acquire and free an area used for user request data if a TYPE=PAGEOUT must be simulated before processing the user's request.*

   f. *Free the returned page list (TYPE=PURGE).*

   g. *Free map copies if TYPE=PAGEOUT and pages were being built in response to TYPE=PAGEBLD requests.*

   h. *Free terminal type parameters (TTPs) (TYPE=PAGEOUT).*

8. *DFHMCP communicates with program control to:*

   a. *Load and delete map sets.*

   b. *Link to the page retrieval program (DFHTPR) to process one or more pages of a message if TYPE=PAGEOUT and CTRL=RETAIN or CTRL=RELEASE.*

   c. *Abnormally terminate a task if uncorrectable errors occur.*

9. *DFHMCP communicates with interval control to:*

   a. *Initiate transaction CSPQ.*

   b. *Obtain the current time of day, which is then used to time-stamp AIDs for routed messages.*

   c. *Initiate transaction CSPS for messages to be delivered at some future time.*

10. *DFHMCP communicates with task control to schedule transaction CSPQ for every terminal that is to receive a routed message to be delivered immediately.*

11. *Transient data control is used to send error and informational messages to the master terminal.*

12. *Route list resolution (DFHRLR) 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.

13. *Non-3270 input mapping (DFHIIP) is used to process TYPE=IN or TYPE=MAP macro instructions for a terminal other than an IBM 3270 Information Display System.*

14. *3270 mapping (DFHM32) is used to process TYPE=IN or TYPE=MAP macro instructions for an IBM 3270 Information Display System.*

15. *Page and text build (DFHPBP) processes all output requests (TYPE=OUT, TYPE=STORE, or TYPE=RETURN). For 3270 output, DFHM32 is called; for other output, DFHML1 is called.*

16. *The partition handling program (DFHPHP) is called when the data is in an inbound structured field. DFHPHP extracts the partition ID, device AID, and cursor address.*

17. *The mapping control program calls DFHMCX if the request is eligible for the BMS fast path module.*

LU1 Printer with Extended Attributes Mapping (DFHML1, BMS)

The LU1 printer with extended attributes mapping program (DFHML1) is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 45 on page 174 shows how the LU1 printer with extended attributes mapping program responds to these requests.

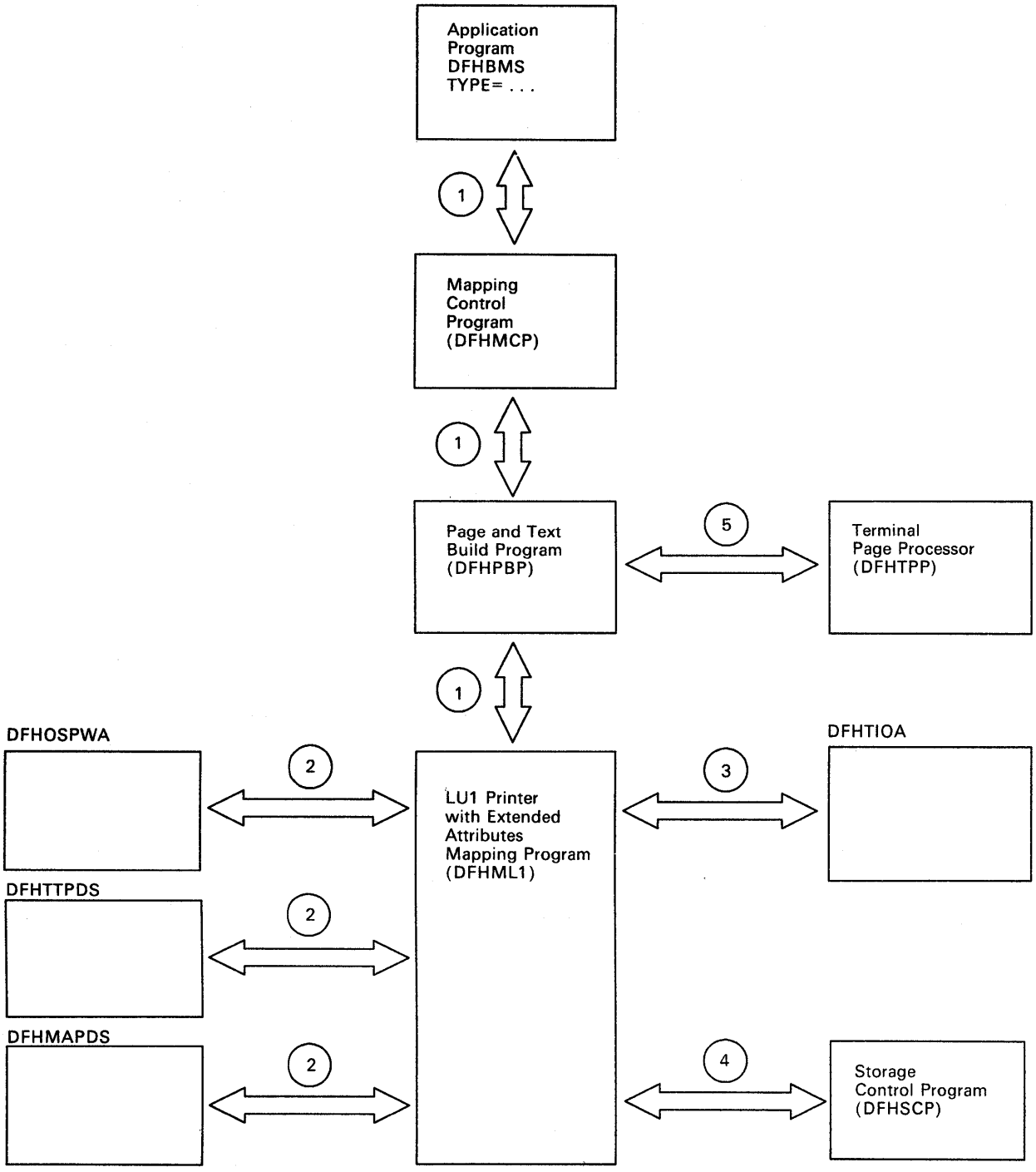


Figure 45. LU1 Printer with Extended Attributes Mapping Program Interfaces

Notes:

1. *A DFHBMS TYPE=PAGEBLD, TEXTBLD, OUT, or RETURN request by an application program communicating with LU1 printer mapping passes information via the TCA through the mapping control program (DFHMCP) and the page and text build program (DFHPBP) to DFHML1.*

   For one page of output, DFHML1 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information on chaining down the maps and processing the fields in each map. DFHML1 then builds the data stream directly from the maps and the TIOAs.

2. *Maps are either passed by the application program or loaded by DFHMCP.*

3. *The address of a terminal input/output area (TIOA) is supplied by the application program for all requests.*

4. *DFHML1 communicates with storage control (DFHSCP) to obtain and release storage for MCAs and for the mapped data.*

5. *All requests (see note 1) are sent to a designated destination by the terminal page processor (DFHTPP), after the return of control to DFHPBP.*

3270 Mapping (DFHM32, BMS)

The 3270 mapping program (DFHM32) is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 46 on page 176 shows how the 3270 mapping program responds to these requests.

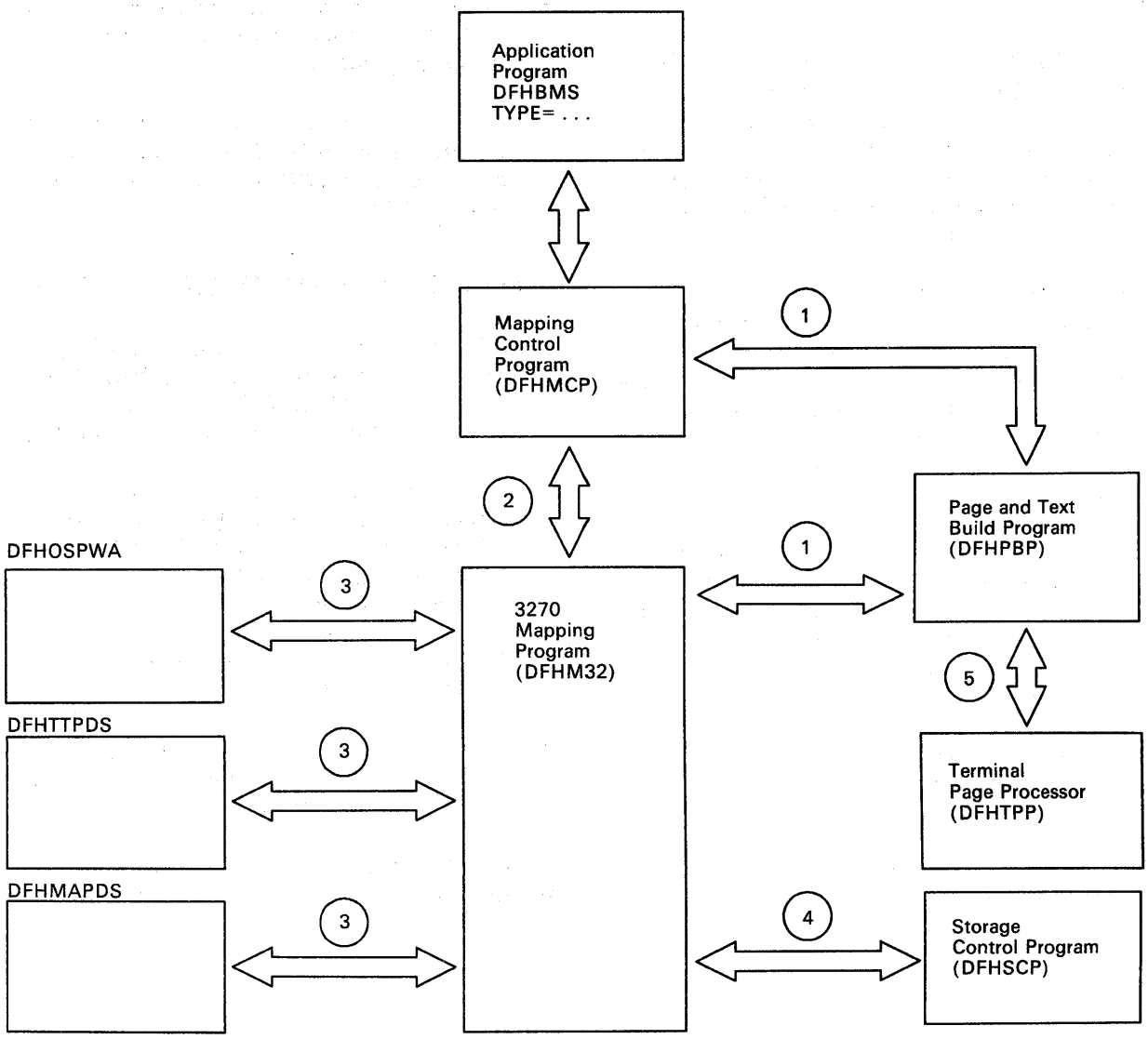


Figure 46. 3270 Mapping Program Interfaces

Notes:

1. *A DFHBMS TYPE=PAGEBLD, TEXTBLD, OUT, STORE, or RETURN macro request by an application program communicating with an IBM 3270 Information Display System passes information via the TCA through the mapping control program (DFHMCP) and the page and text build program (DFHPBP) to DFHM32.*

   For one page of output, DFHM32 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information for chaining down the maps and processing the fields in each map. DFHM32 then builds the data stream directly from the maps and the TIOAs.

2. *A DFHBMS TYPE=IN or TYPE=MAP macro request by an application program communicating with an IBM 3270 Information Display System passes information through the TCA through the message control program (DFHMCP) to DFHM32.*

3. *Maps are either passed by the application program or loaded by DFHMCP.*

4. *DFHM32 communicates with storage control (DFHSCP) to obtain and release buffers for mapping operations.*

5. *All output requests (see note 1) are sent to a designated destination by the terminal page processor (DFHTPP) after control is returned to DFHPBP.*

Page and Text Build (DFHPBP, BMS)

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 an IBM 3270 Information Display System (TYPE=PAGEBLD).
- Places the data into the page buffer (TYPE=TEXTBLD).
- Inserts device dependent control characters for other than 3270 Information Display System devices, removing extended attributes.

Figure 47 on page 178 shows the relationships between the components of page and text build.

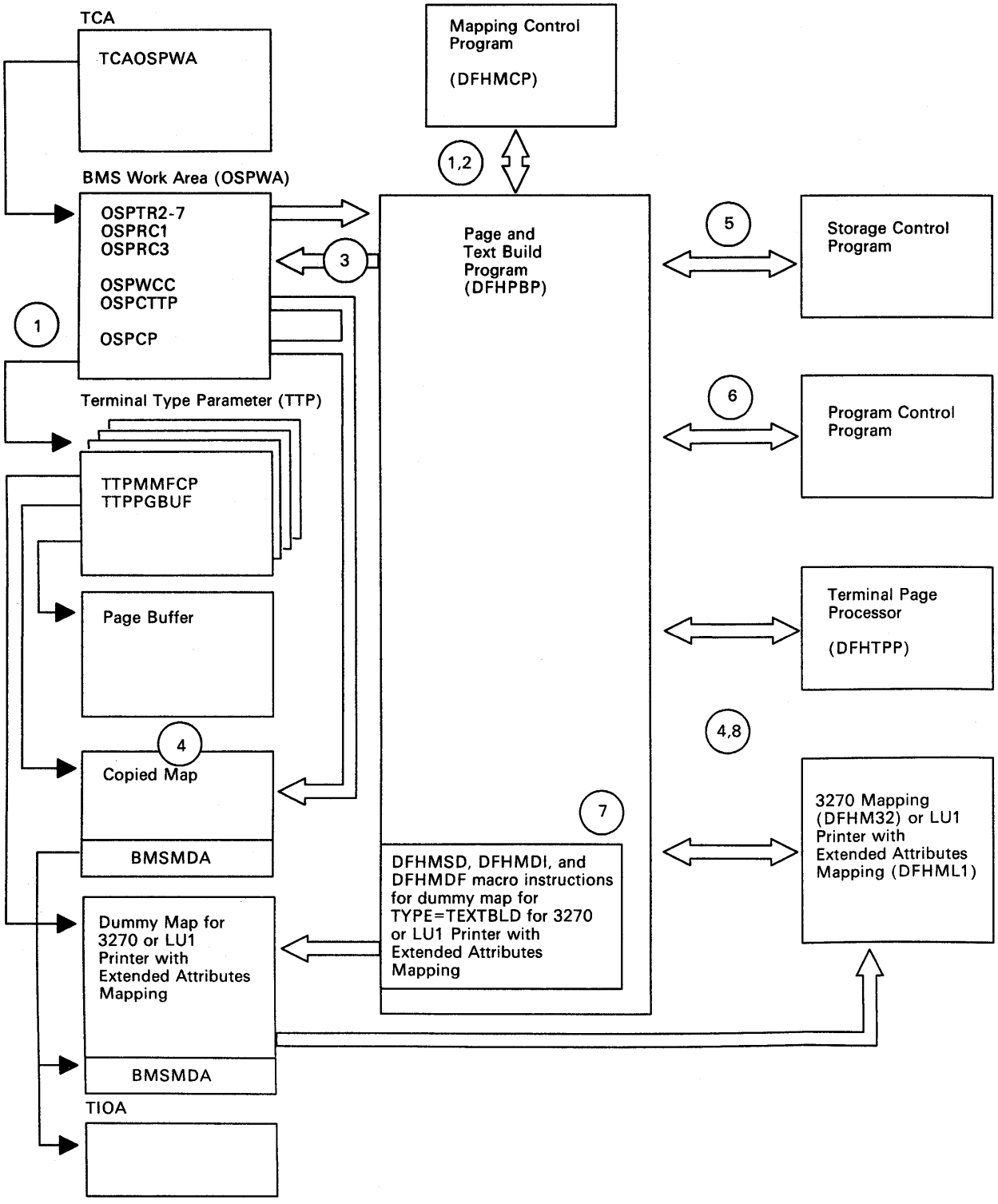


Figure 47. Page and Text Build Program Interfaces

Notes:

1. *DFHPBP is entered from the mapping control program (DFHMCP) to process all BMS output requests. It is called once for each terminal type parameter (TTP) on the TTP chain pointed to by OSPTTP. The current TTP in the chain is pointed to by OSPCTTP.*

2. *DFHPBP returns control to DFHMCP 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.*

3. *OSPTR2, OSPTR3,...,OSPTR7 contain request data from the DFHBMS macro expansion. OSPRC1 and OSPRC3 contain return codes to be examined by DFHMCP.*

4. *For a TYPE=PAGEBLD request for an IBM 3270 Information Display System, the map is copied and chained to the TTP. For a TYPE=TEXTBLD request for an IBM 3270 Information Display System, a dummy map is created and chained to the TTP. When a page is complete, control is given to 3270 mapping (DFHM32), which combines the map copies chained to the TTP and maps the data.*

   For a TYPE=PAGEBLD request for an LU1 printer with extended attributes, the map is copied and chained to the TTP. For a TYPE=TEXTBLD request, a dummy map is created and chained to the TTP. When a page is complete, control is given to the LU1 printer mapping program (DFHML1), which combines the map copies chained to the TTP and maps the data.

5. *DFHPBP communicates with storage control to*

   a. *Acquire and free buffers in which pages are built.*

   b. *Acquire storage for copies of maps for TYPE=TEXTBLD or TYPE=PAGEBLD.*

   c. *Acquire storage for a copy of the user's data for TYPE=TEXTBLD or TYPE=PAGEBLD.*

6. *DFHPBP requests program control to abnormally terminate a transaction (DFHPC TYPE=ABEND) if certain uncorrectable errors occur.*

7. *TYPE=TEXTBLD request for an IBM 3270 Information Display System causes a map set consisting of one dummy map to be passed to 3270 mapping (DFHM32). The map has one field with attributes FREEKB and FRESET.*

   TYPE=TEXTBLD requests for an LU1 printer causes a map set consisting of one dummy map to be passed to the LU1 printer mapping program (DFHML1). The map has one field with attributes FREEKB and FRESET.

8. *If the page is being constructed for an IBM 3270 Information Display System, control is given to DFHM32 to map the data and then to DFHTPP to output the page.*

   If the page is being constructed for an LU1 printer, control is given to DFHML1 to map the data, and then to DFHTPP to output the page. Otherwise, control is given to DFHDSB to add device dependencies to the page, and then to the terminal page processor (DFHTPP) to output the page.

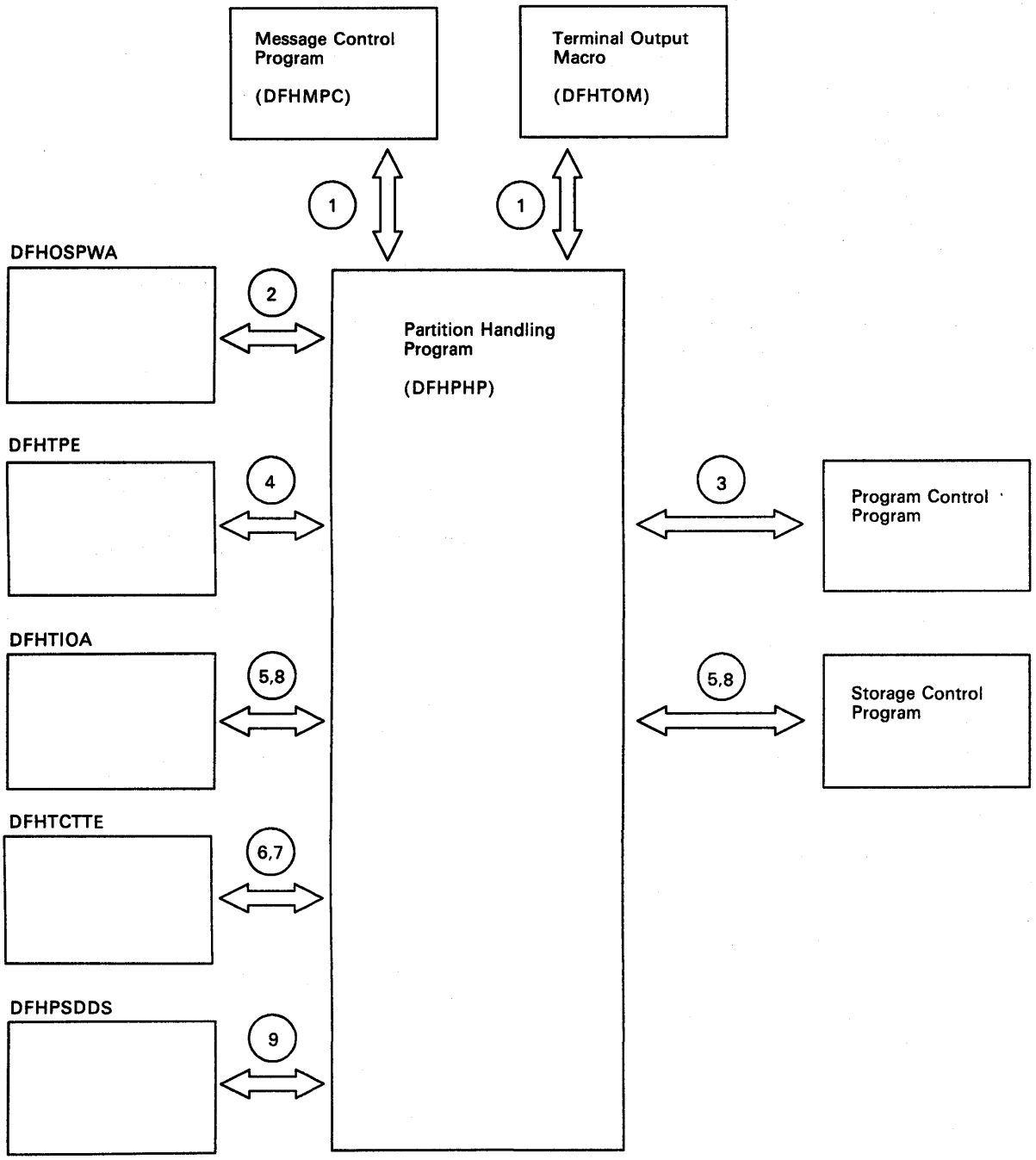


Figure 48. Partition Handling Program Interfaces

Partition Handling Program (DFHPHP, BMS)

The partition handling program (DFHPHP) processes terminal operations that involve partitions. DFHPHP has one entry point, and starts with a branch table which passes control to the required routine according to the request. It consists of routines that perform the following functions:

The routine PHPPSI tests whether there is a partition set in storage. If there is and it is not the required partition set, then that partition set is deleted. When no partition set is in storage, an attempt is made to load the appropriate partition set.

PHPPSC builds a data stream to destroy any partitions that may already be loaded on the terminal, creates the partition set designated by the application partition set, and sets the name of the partition set in the TCTTE to be the name of the application partition set.

PHPPIN extracts the AID, cursor address, and partition ID. The AID and cursor address are put in the TCTTE, and the partition ID is converted to a partition name and returned to the caller. A check is made that the partition ID is a member of the application partition set.

PHPPXE sends a data stream to a terminal to activate the appropriate partition and sends an error message to any error message partition if input arrived from a partition other than the expected input partition.

Figure 48 on page 180 shows the relationships between the components of partition handling.

Notes:

1. *DFHPHP is called by the mapping control program (DFHMCP) and by the terminal output macro (TOM).*
2. *PHPPSI refers to OSPWA to check if a partition set is loaded.*
3. *PHPPSI communicates with program control to load the partition set.*
4. *PHPPSI puts the name of the partition set in TPE (terminal partition extension) as the application partition set.*
5. *PHPPSC calls storage control to acquire a TIOA in which to build and free the original TIOA.*
6. *PHPPSC sets a slot in TCTTE to be the partition set data stream concatenated with terminal partition set name if the terminal is not in the base state.*
7. *PHPPIN places the AID and the cursor address in the TCTTE.*
8. *PHPPXE calls storage control to get a TIOA, calls DFHMGP to get the error message text, fills the TIOA with data, transmits the data, and frees the TIOA.*
9. *PHPPSC references the partition set object to build the partition creation data stream.*

Route List Resolution Program (DFHRLR, BMS)

The route list resolution program (DFHRLR) builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.

Figure 49 on page 182 shows route list resolution program interfaces.

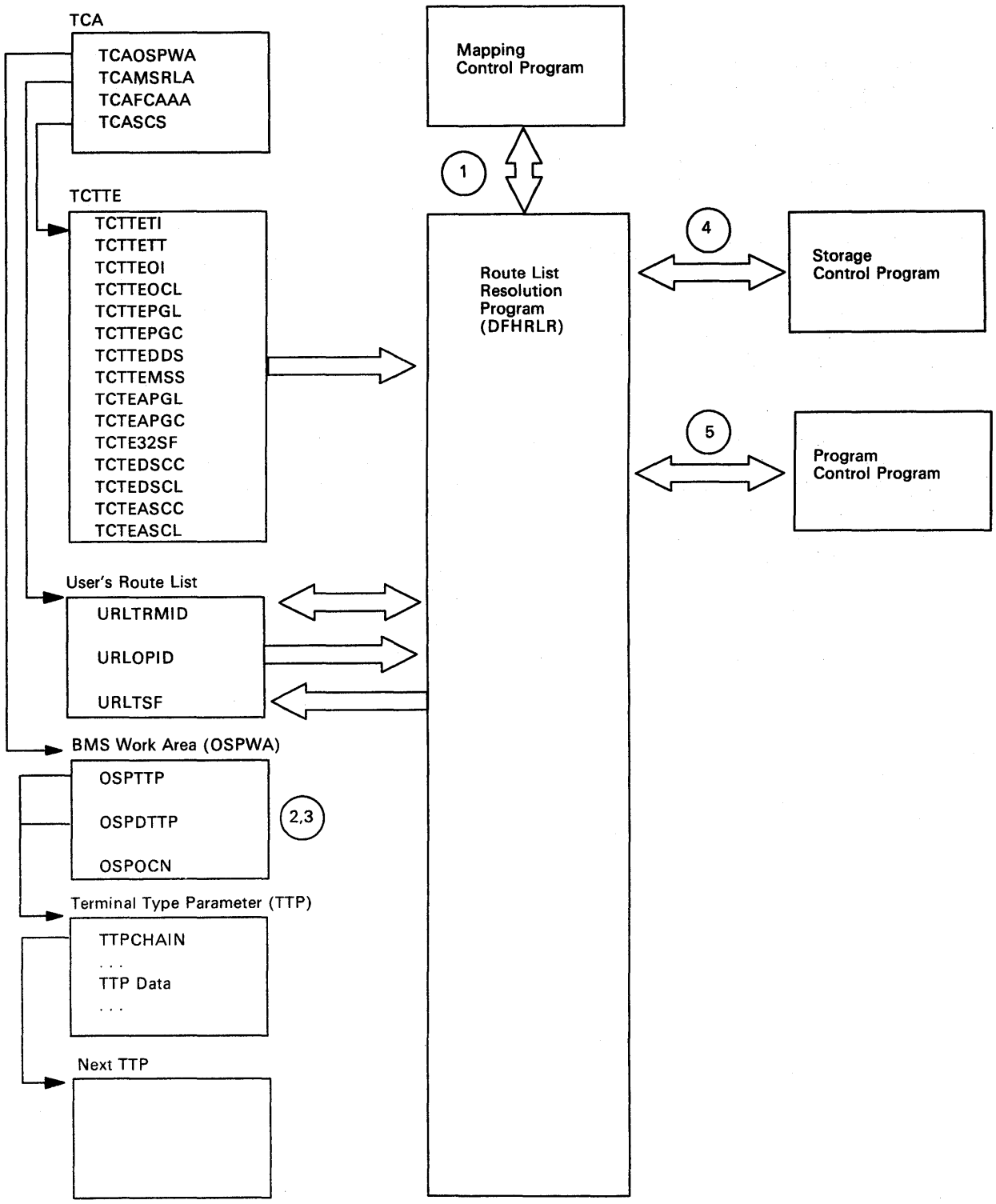


Figure 49. Route List Resolution Program Interfaces

Notes:

1. *DFHRLR is called by the mapping control program (DFHMCP) to determine the grouping of terminal destinations.*

2. *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. For each TTP, the supported attributes of the corresponding terminals are accumulated. The address of the first routing TTP in the chain of TTPs is placed in OSPTTP.*

3. *If data is not to be routed, a direct TTP is built for the originating terminal and its address is placed in OSPDTTP.*

4. *DFHRLR communicates with storage control to acquire storage for the TTP.*

5. *Program control services are requested by means of a DFHPC TYPE=ABEND macro instruction if certain uncorrectable errors occur.*

Terminal Page Processor (DFHTPP, BMS)

The terminal page processor (DFHTPP) puts completed pages to a destination specified in the BMS output request (TYPE=OUT sends to the originating terminal, TYPE=STORE directs to temporary storage, and TYPE=RETURN directs to a list of completed pages that are returned to the application program).

Figure 50 on page 184 shows the relationships between the terminal page processor and other components in response to BMS output requests.

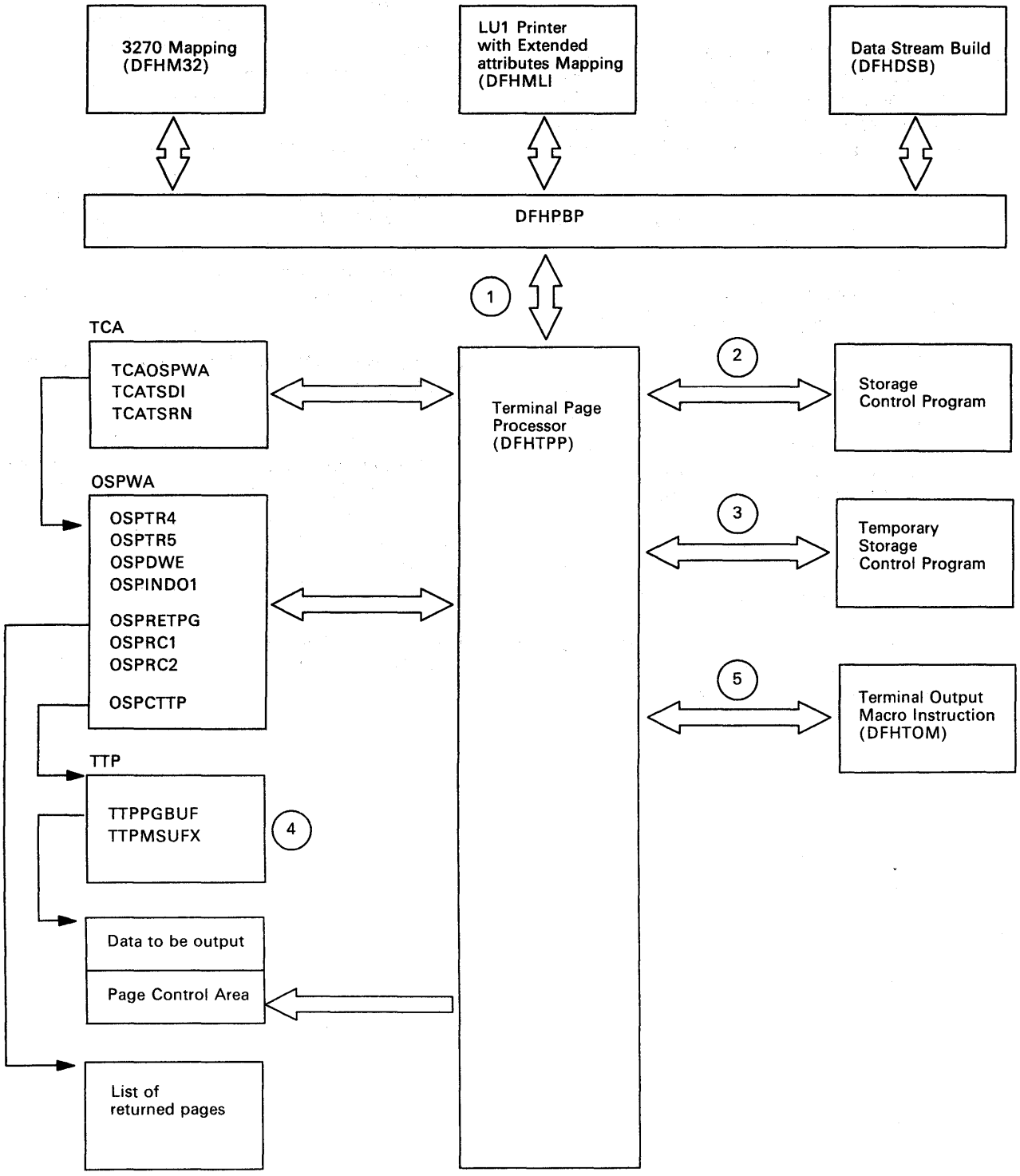


Figure 50. Terminal Page Processor Interfaces

Notes:

1. *DFHTPP is entered from DFHPBP after processing by 3270 mapping (DFHM32) for 3270s, by LU1 printer with extended attributes mapping (DFHML1) for those LU1 printers, and by data stream build (DFHDSB) for other devices.*

2. *DFHTPP communicates with storage control to obtain.*

   a. *The return list (to store the address of completed pages to be returned to the programmer).*

   b. *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.*

   c. *Storage for a list that correlates pages on temporary storage with the logical device codes for which they are destined.*

3. *Temporary storage control is used to store pages and the message control record (MCR) for messages stored on temporary storage.*

4. *The terminal type parameter (TTP) controls the formatting of a message for a particular terminal type, for example, an IBM 2741 Communications Terminal. TTPPGBUF contains the address of a completed page.*

5. *The terminal output macro instruction (DFHTOM) is issued to provide an open subroutine assembled within DFHTPP that puts a completed page out to the terminal. If the data stream contains extended attributes, and the terminal does not support extended attributes, the extended attributes are deleted.*

Undelivered Messages Cleanup Program (DFHTPQ, BMS)

The undelivered messages 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).

Figure 51 on page 186 shows the undelivered-messages-cleanup program interfaces.

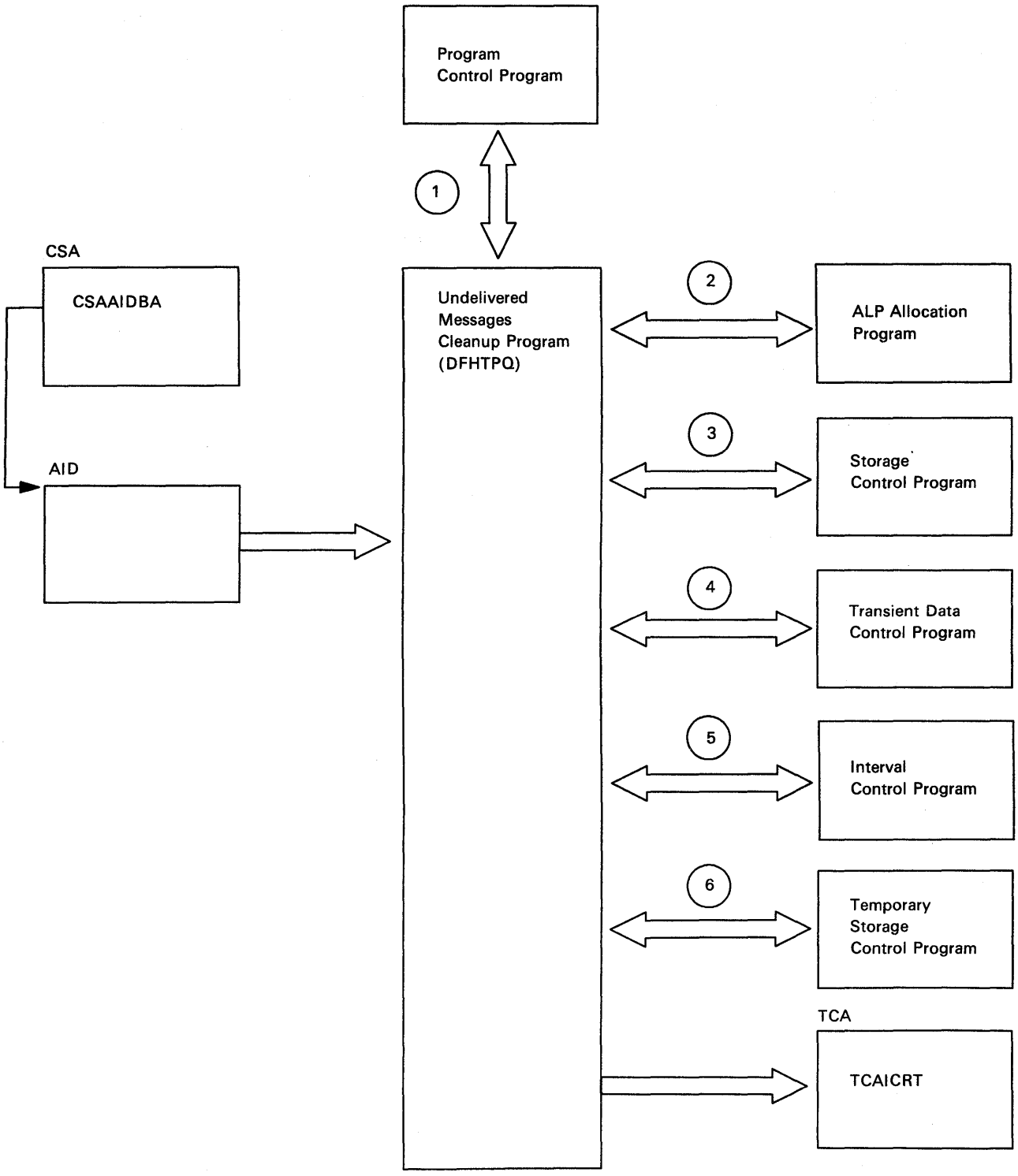


Figure 51. Undelivered Messages Cleanup Program Interfaces

Notes:

1. *DFHTPQ is initiated the first time by the mapping control program (DFHMCP) by interval control or by the transaction CSPQ. Thereafter, it reinitiates itself (see note 5).*
2. *DFHTPQ communicates with the allocation program (DFHALP) to locate and unchain AIDs.*
3. *DFHTPQ communicates with storage control to free AIDs which have been purged and to acquire storage for notification messages.*
4. *Transient data control is used to send notification messages.*
5. *Interval control is used to obtain the current time and to reinitiate this task (DFHTPQ).*
6. *DFHTPQ communicates with temporary storage control to retrieve and replace message control records (MCRs) and to purge messages.*

Page Retrieval Program (DFHTPR, BMS)

The page retrieval program (DFHTPR) processes messages built by BMS and placed in temporary storage.

Figure 52 on page 188 shows the relationships between the components of page retrieval.

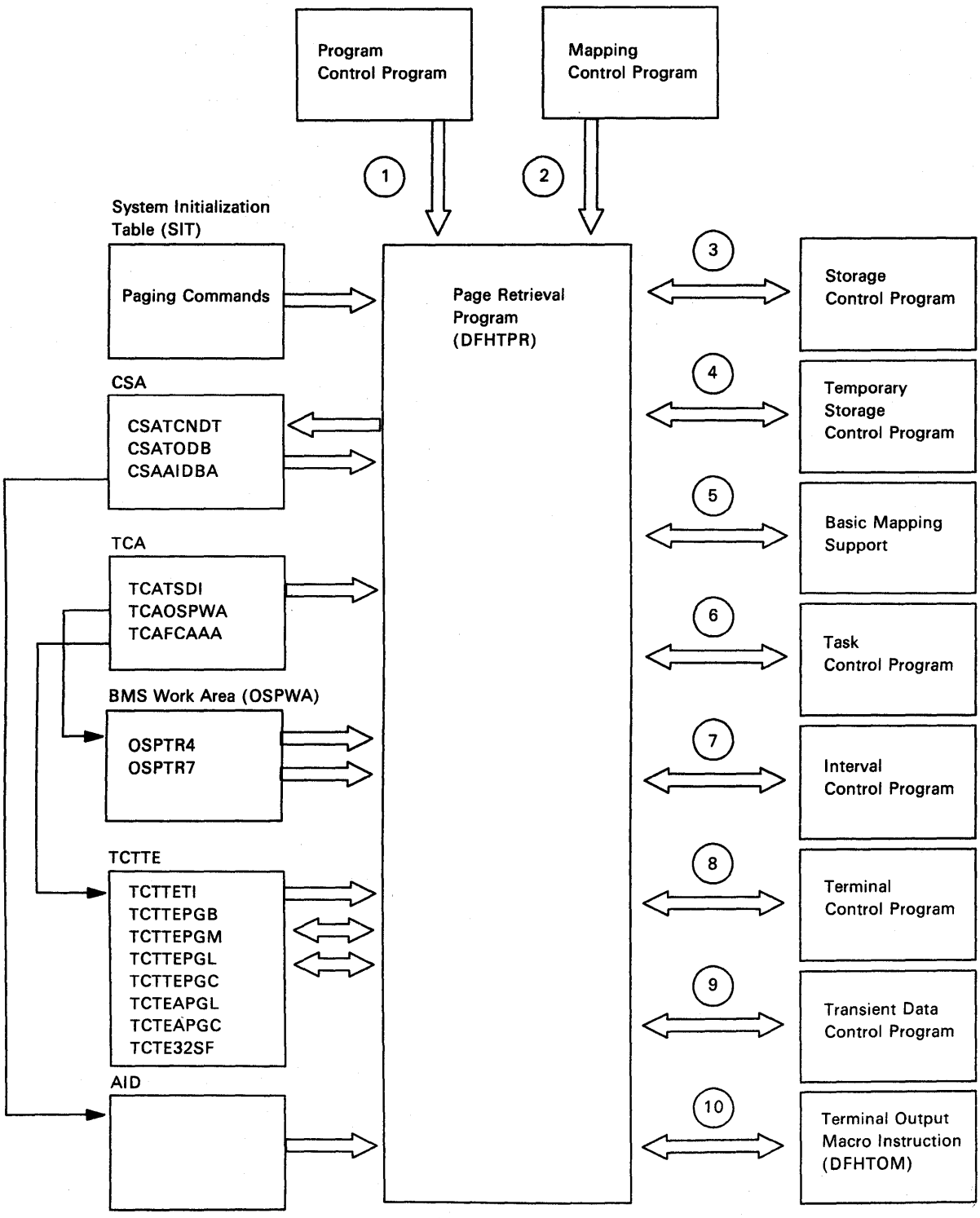


Figure 52. Page Retrieval Program Interfaces

Notes:

1. *DFHTPR can be initiated as a stand-alone transaction (CSPQ or user defined paging command, for example, P/, or 3270 PA/PF keys), or linked to from BMS conversational operation. (DFHBMS TYPE=PAGEOUT, CTRL=RELEASE\|RETAIN). The functions of DFHTPR are:*

   a. *Display the first page of a routed message.*

   b. *Display subsequent pages of a message at a terminal for which TYPE=PAGEOUT, CTRL=AUTOPAGE was specified.*

   c. *Process paging commands from a terminal.*

   d. *Process the transaction CSPG when it is entered at the terminal.*

   e. *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.*

2. *DFHTPR is entered from the BMS mapping control program (DFHMCP) to display the first page of a message originated at the terminal if CTRL=RETAIN was specified in the BMS request. DFHTPR reads from the terminal and processes paging commands until other than a paging command is entered.*

3. *DFHTPR uses storage control to:*

   a. *Acquire and free message control blocks (MCBs).*

   b. *Free message control record (MCR) storage.*

   c. *Acquire storage for informational and error messages to be sent to the destination terminal and the master terminal.*

   d. *Free an automatic initiate descriptor (AID) taken off the AID chain.*

   e. *Acquire and free storage for a route list constructed in response to a COPY command entered at a terminal.*

   f. *Acquire a TIOA into which to place a device-independent page when performing the COPY function.*

4. *Temporary storage control is used to retrieve and replace MCRs and to retrieve and purge pages.*

5. *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.*

6. *Task control is used to retain exclusive control of a MCR while it is being updated.*

7. *DFHTPR communicates with interval control 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.)*

8. *Terminal control 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.*

9. *Transient data control is used to send error or informational messages to the master terminal.*

10. *The terminal output macro instruction (DFHTOM) is issued to provide an open subroutine which puts a completed page out to the terminal.*

Terminal Page Scheduling Program (DFHTPS, BMS)

The terminal page scheduling program (CSPS) is invoked for each terminal type to which a BMS logical message built with TYPE=STORE is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

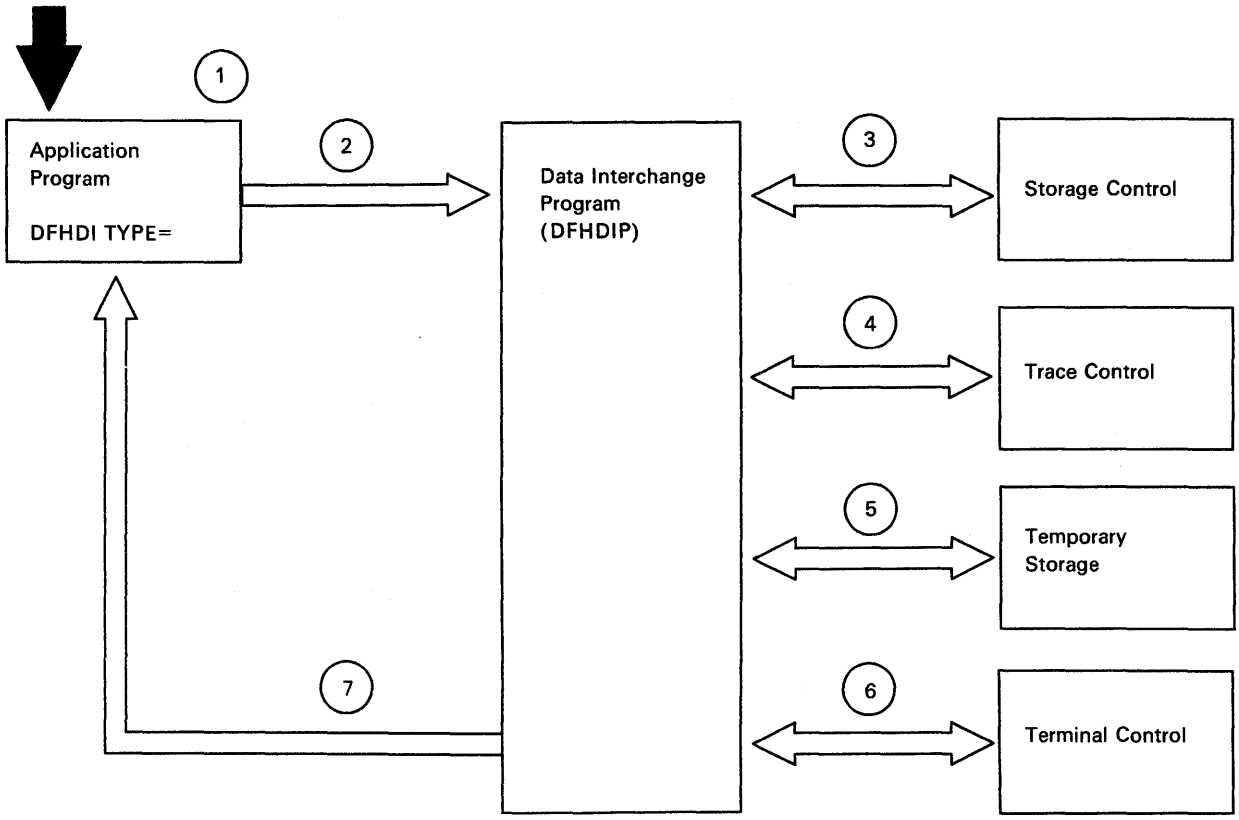


Figure 53. Data Interchange Program

Data Interchange Program (DFHDIP)

The data interchange program is designed as a function manager for SNA devices. It is invoked by DFHDI requests directly or by the BMS routines from DFHBMS requests (that is, the BMS routines issue DFHDI requests). DFHDIP provides the following functions:

1. Determines whether a new output destination has been specified and, if so, builds appropriate FMHs to select the new destination, and outputs these FMHs to the SNA device via terminal control.

2. Invokes the appropriate subroutine to perform the desired function:

   - ADD – build ADD FMH, transmit it and the user data
   - REPLACE – build REPLACE FMH, transmit it and the user data
   - ERASE – build ERASE FMH, RECID FMH and transmit it
   - NOTE – build NOTE FMH, transmit it, return the reply to the user
   - QUERY – build QUERY FMH, transmit it, and output END FMH
   - SEND – output user data
   - WAIT – wait for completion of the I/O
   - END – build END FMH and transmit it
   - ABORT – build ABORT FMH and transmit it
   - ATTACH – remove FMH from initial input
   - DETACH – free the storage used by DFHDIP
   - RECEIVE – read a complete record from the logical device.

3. Sets the appropriate return code.

Figure 53 shows the relationships between the components of data interchange.

Notes:

1. *The application program issues DFHDI requests.*
2. *DFHDIP receives control.*
3. *If no storage has been obtained for the data interchange block (DIB) then storage control is invoked. The storage is chained to the TCTTE.*
4. *A trace entry is made.*
5. *If logging is present (protected task and message integrity) and if a destination change or function change occurs on output then temporary storage is invoked to write the DIB to recoverable temporary storage.*
6. *Terminal control is invoked to output any built FMH and also to output the user data. (DFHTC TYPE= WRITE is issued). For input requests DFHTC TYPE= READ requests are issued to obtain a nonnull input record.*
7. *Any errors obtained from the device are decoded and placed in the TCA return code slot. If no errors were detected then a return code of zero is returned.*

Built-In Functions

Several commonly used functions are available to the application programmer through the use of CICS 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) includes table search, phonetic conversion, field edit, field verify, bit manipulation, input formatting, and weighted retrieval. 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 allows the application program to search 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 allows the CICS user to convert 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 subroutine to convert keys in a similar manner is provided for use by offline programs. Together, these facilities allow the CICS user to organize files of names so that they can be accessed by names that may be misspelled, mispronounced, or misunderstood.

You can write names prefixed by "Mc" with or without a blank between the prefix and the rest of

the name, for example, "McEWEN" or "Mc EWEN".

Field Verify/Edit

The field verify function enables CICS 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:

- Entirely alphabetic: blanks or A-Z
- Entirely EBCDIC digits: 0-9
- Entirely packed decimal: (COMPUTATIONAL-3 in 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 nonnumeric 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 key-sequenced 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.

Execution (Command-Level) Diagnostic Facility

The execution diagnostic facility (EDF) enables an application programmer to test a command-level application program online without making any modifications to the source program or the program preparation procedure. EDF intercepts execution of the application program at certain points and displays relevant information about the program at these points.

EDF debug mode is switched on and off by a transaction or PF (program function) key named in the PCT by the system programmer; also, the PPT needs to specify the programs and maps that are used by EDF. EDF uses temporary storage and simple BMS; the system programmer must make sure that these facilities are available.

Note: The PCT entry for CEDF has the default value TRACE = NO which inhibits trace entries while EDF is running. The PCT entry should be

specified as TRACE = YES if trace entries are required.

Command Interpreter

The command interpreter provides an interactive, display-oriented tool to help in the writing, syntax checking, and execution of CICS commands. Commands can be entered completely or partially with prompts for the remaining options. The command interpreter displays any syntax errors and the results of execution of the command.

The command interpreter consists of two parts – the display component and the translator component. The display component presents information to the user on a display and receives commands and display instructions from the user. The translator component is similar to the command translator, and uses the translator tables which are common to both.

Temporary Storage Browse Transaction

The temporary storage browse transaction allows the user to browse, copy, or delete items in a queue. The browse is performed by requesting the CEBR transaction. CEBR invokes DFHEDFBR to execute the required action.

You can use the CEBR transaction to copy transient data queues to temporary storage, but you cannot read an output extrapartition transient data queue.

Chapter 2.7. Extended Recovery Facility

This chapter contains the following sections:

- "General introduction"
- "Elements of XRF"
- "Task control blocks (TCBs)"
- "Initialization"
- "CICS availability manager"
- "CICS availability manager SVC services"
- "XRF terminal control"
- "The XRF overseer."

General introduction

This chapter is a functional description of a CICS/MVS system running with XRF = YES specified in the system initialization table (SIT). For information about functions common to systems running with XRF = YES and XRF = NO, refer to the relevant sections of this manual.

The *CICS/MVS XRF Guide* gives an overview of the way an XRF system works, but, before the functional description proper begins, the next few pages contain an overview of an XRF system. If you are already familiar with this description from another manual, move to "Elements of XRF" on page 205.

The following discussion is mainly for systems in a single region.

Overview of MVS/CICS running as an XRF system

CICS/MVS in XRF mode is a system approach to increased availability. It uses alternate resources to minimize hardware and software outages – both planned and unplanned.

XRF involves a pair of CICS systems:

- The **active system** running the CICS workload
- The partially initialized **alternate system**, standing by in case of failure.

This partially initialized alternate system, probably on a separate central electronic complex (CEC), enables you to provide greater availability to your end users. There is more protection if you use two CECs, but the two systems can be in the same CEC. It can do this by reacting automatically to problems that cause interruptions to service. Through the **CICS availability manager (CAVM)**, the alternate system constantly communicates with the working, or active, CICS system, recording changes in terminal usage – **tracking** – and monitoring the well-being of the active system – **surveillance**. Surveillance and tracking information is passed through the CAVM data sets – the **message data set** and the **control data set**. These data sets are on shared DASD, accessible to both active and alternate systems. When the alternate system detects a failure in the active system, or when it is instructed to act, it has access to all the necessary information and resources to **take over** from the active system and reestablish service with the minimum of interruption.

When the alternate system takes over the running, it performs an emergency restart similar to an emergency restart after the failure of a non-XRF CICS system. With XRF, the whole emergency restart process is faster. This is because the

alternate system is already partially initialized, with backup VTAM sessions from the 372x to the alternate system ready for its terminals, and because the restart is initiated sooner due to the surveillance activity.

Note: Most of your existing emergency restart procedures remain valid for XRF, because XRF builds on the existing CICS emergency restart.

The alternate system is only partially initialized. It cannot complete its initialization until its active partner has terminated. It cannot carry out any normal processing until it has taken over and become the new active system. Partial initialization has the advantage that the alternate system takes up very little resource, and leaves your second CEC available for other work.

Terminal capability

Terminals that are XRF-capable, that is, have backup sessions, benefit from the improved restart after a failure.

SNA VTAM terminals connected to a 3270 Information Display System, or a boundary network node 3725 Communications Controller that is controlled by an NCP with XRF capability, can be switched to the alternate system, and retain their sessions. This is made possible by the NCP in the 3725 Communications Controller, which creates backup sessions to the alternate system for all terminals with backup sessions. Other VTAM terminals can be tracked, and acquired by the alternate system after takeover. These terminals benefit from improved restart after a failure. Some terminals might need manual intervention to switch to the alternate system.

Causes of a takeover

The takeover might occur because of a failure in the active system, the CEC, MVS/XA, or VTAM, or the system operator can initiate a planned takeover. Once a failure has occurred, and the alternate system has become the active system, you can initialize another alternate system, and maintain the extended recovery facility. To make changes to your CICS system, you can initiate takeover to an alternate system that has incorporated software maintenance, or a changed configuration. That alternate system becomes the new active system, and then you can back it up with a new alternate system, which can be in the old active CEC.

Brief description of the way an XRF system works

Before CICS/MVS Version 2, a CICS failure meant that you needed to restart your system, probably using an emergency restart. An XRF takeover, which is an enhanced emergency restart, provides the same **integrity** as an emergency restart in a non-XRF system. To the end user, the takeover has a similar appearance to an emergency restart. Most of your existing emergency restart procedures will remain valid for XRF. However, if in the past you have delayed the restart to allow (for example) post-processing or pre-processing job steps to be carried out, this will not be possible in an XRF takeover, which does not allow for this sort of delay.

Figure 54 on page 197 shows a possible sequence of XRF operations. In the next five sections, these five stages in the sequence are described:

1. Initialization
2. Synchronization
3. Surveillance and tracking
4. Takeover
5. After takeover.

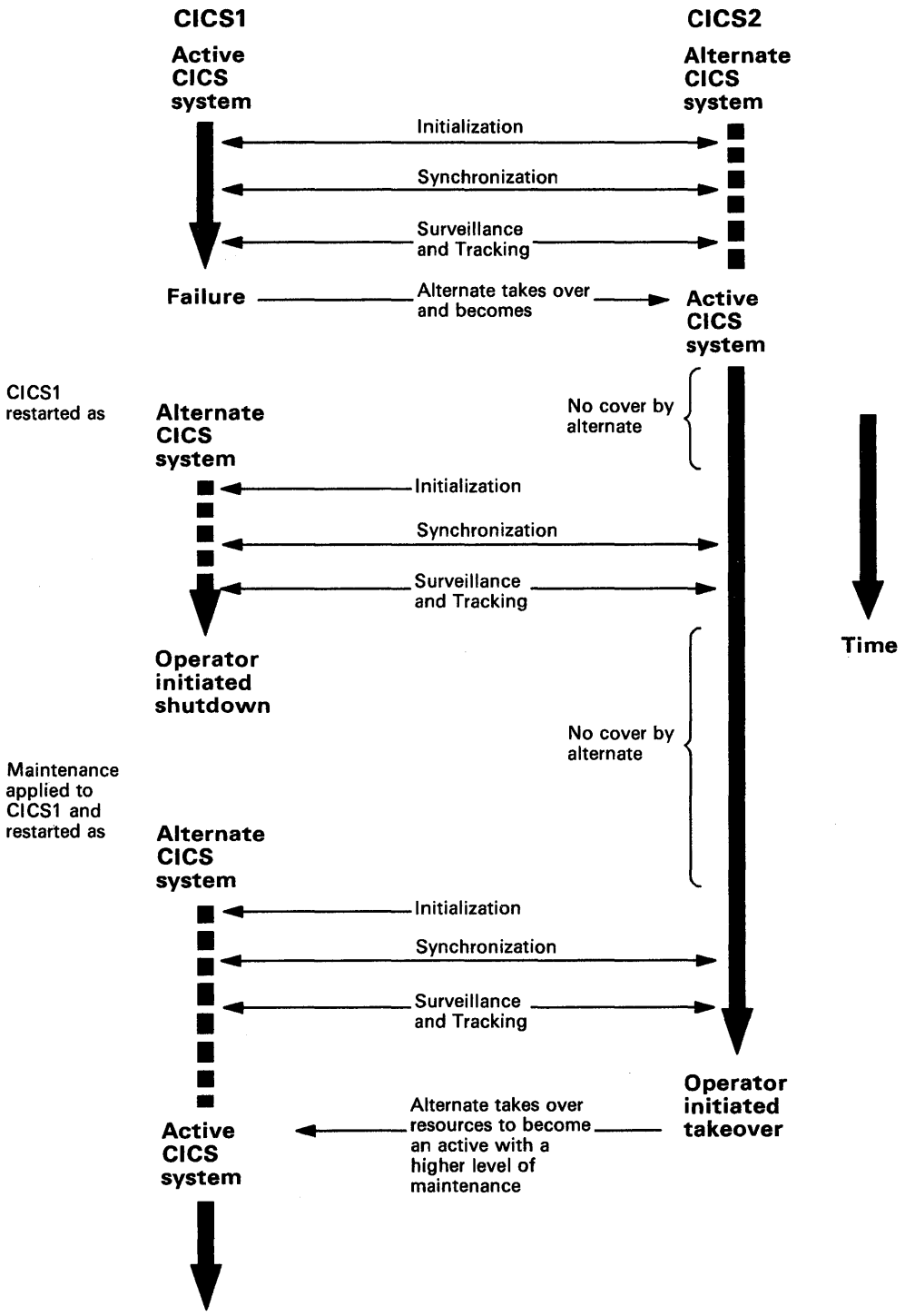


Figure 54. An XRF sequence

Initialization

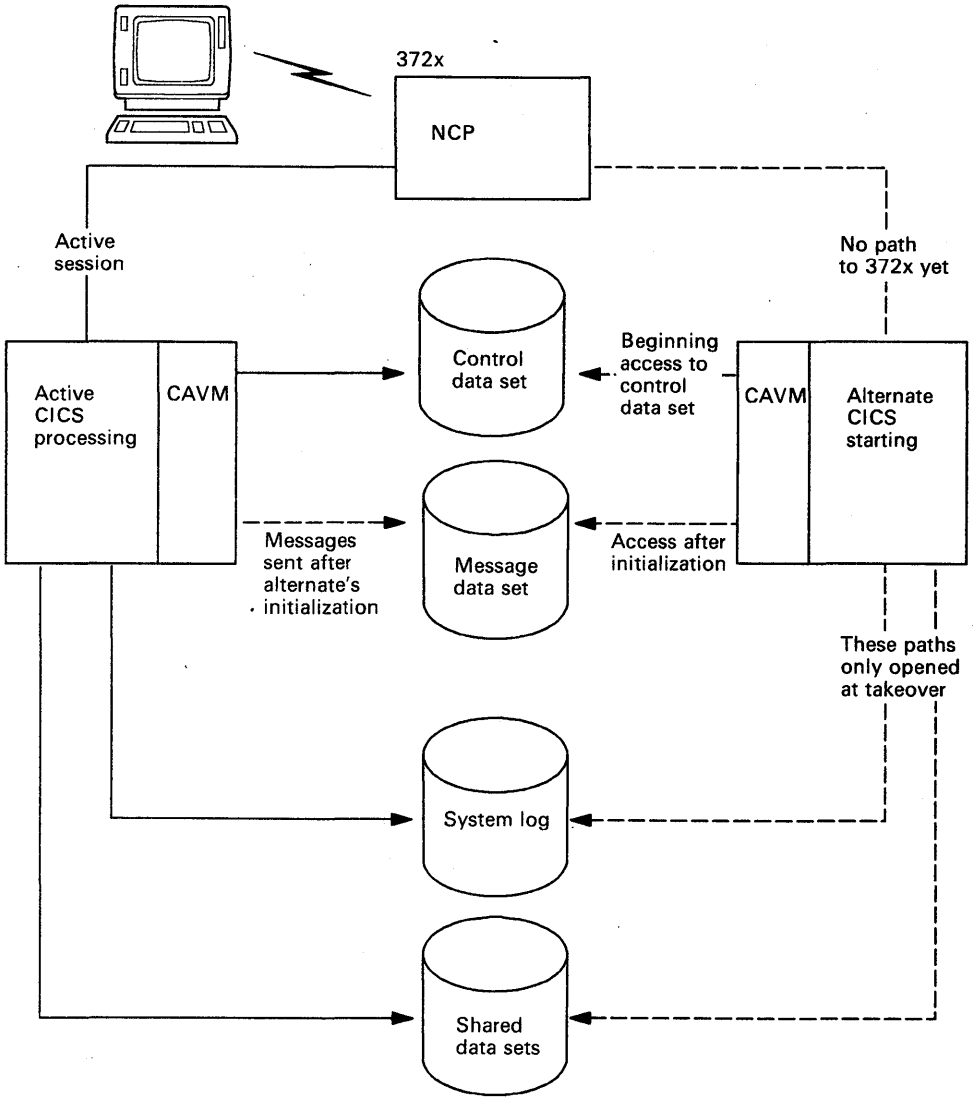


Figure 55. Initialization of the alternate system after the active system has started processing

To use XRF, you need a pair of CICS systems - the active system and the alternate system. You start the active system and the alternate system separately, and you can start them concurrently, or in either order. The start-up job streams for active and alternate systems (there are some examples in the *CICS/MVS Operations Guide*) must be very similar, except for some of the SIT parameters (probably overrides), and certain data-set definitions. The active and alternate systems have their own dump, auxiliary trace, and extrapartition transient-data data sets. Apart from such minor differences, the active and alternate systems must be compatible, with the same recoverable resource definitions. This ensures that, after takeover, the new active system provides the same service as before.

The active and the alternate systems **sign on** to the CICS availability manager (CAVM) at the start of initialization.

The CAVM is the mechanism which enables active and alternate systems to coordinate their processing. The CAVM uses a shared pair of data sets: a control data set and a message data set.

This pair of data sets is logically a single entity which contains:

- State data whose main purpose is to ensure that, at any given time, at most one of the CICS jobs sharing that particular pair of data sets is allowed to perform the active role.
- Primary and secondary surveillance signals of active and alternate systems, so that each system can tell whether its partner is functioning properly.
- Messages about the state of certain resources in use on the active system, which are written by the active system, and read and processed by the alternate system.

CAVM rejects a request from a CICS job to sign-on as the active system if the control data set shows that an active CICS is already present, or that a takeover is in progress. This ensures that the integrity of files and data bases cannot be lost as a result of uncontrolled concurrent updating by two or more active systems. As soon as an active or alternate system signs on, it starts to write its own surveillance signals, and to look for its partner's surveillance signals.

The control data set is used:

- To record the presence or absence, identities, and current state of active and alternate CICS jobs.
- For the primary surveillance signals of the active and the alternate systems.

The message data set is used:

- Principally to pass messages about the current state of certain resources from the active system to the alternate system.
- For the secondary surveillance signals of the active and/or alternate systems, when the control data set is unavailable for this purpose, either because the last write has not completed yet, or because of I/O errors.

For more details about the CAVM data sets, see the *CICS/MVS Operations Guide*.

The active system completes its initialization normally. It then begins to provide a service to its end users.

The alternate system cannot be fully initialized because, until it takes over from its active counterpart, it does not own the resources that can be used by only one system at a time, such as the system log and user data sets. The alternate system is initialized only to the point at which it can monitor the active system. VTAM must be running before the alternate system can complete its initialization. Only one alternate system at a time is allowed to sign on to the CAVM. If the alternate system is started first, it just waits, watching for its active partner's surveillance signals to start when it signs on to the CAVM.

The alternate system cannot perform any active CICS function (users cannot log on to it, for example), and it takes up very little resource. The only means of external communication with the alternate system is the MVS MODIFY command, which in turn is limited to a small set of CEBT commands. The alternate system performs **surveillance** and **tracking**, writing its own surveillance signals, reading the active system's surveillance signals, and reading messages describing the status of terminals in the active system.

Synchronization

When the active system is initialized, and it detects that the alternate system has signed on to CAVM, they are at the synchronization stage. The active system uses CAVM message services to send a stream of messages describing the current state of all its VTAM terminals using the message data set to the alternate system. This is called the **catch-up** process, which enables the alternate system to build a complete picture of the active system's terminal resources and the status of those terminals. In this way, the alternate system is aware of the existing terminal network, can request backup sessions for XRF-capable terminals, and can track any remaining VTAM terminals.

If the alternate system stops for any reason, and the active system runs by itself for some time before another alternate system is started, the same catch-up process is performed for the new alternate system.

Then the system enters the surveillance and tracking stage.

Surveillance and tracking

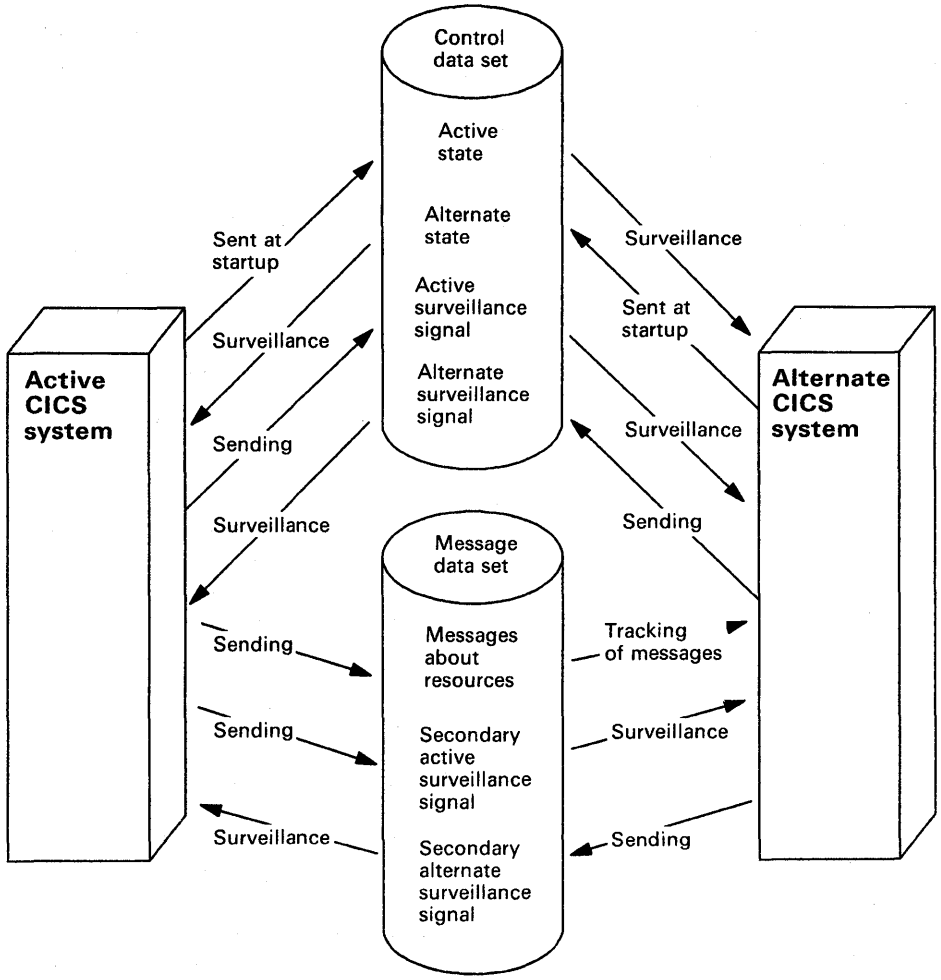


Figure 56. Use of the CAVM data sets for surveillance and tracking

Most of the time, CICS with XRF is in this third stage surveillance and tracking.

The active system sends out surveillance signals to the alternate system, and the alternate system monitors them, checking for any sign of failure in the active system. If the active system itself detects a failure which prevents it from continuing to provide a service, it signs off abnormally from the CAVM to inform the alternate system of its failure. A CEC or operating system failure, or a serious CICS failure causes the active's surveillance signals to stop.

While running normally, the active system uses CAVM message management services to inform the alternate system about changes made to the terminals installed in the system. The active system also informs the alternate system of changes to the logged-on/logged-off state of all VTAM terminals and sessions as they are acquired or released. In this way, the alternate system tracks the

installed/logged-on/logged-off state of all VTAM terminals. If the tracked logged-on state indicates XRF capability, the alternate system requests VTAM to set up a backup session for that terminal so that it can be switched at takeover. This session is set up between the alternate system and the network control program in the 372x.

The emphasis in surveillance is that the alternate system monitors the state of the active system. But the active system continually checks the alternate system's status and its surveillance signals, to make sure that there is an alternate system to receive the messages it is sending. If the alternate system's surveillance signal disappears, or it signs off abnormally from the CAVM, the active system warns the system operator. Loss of the alternate system does not affect the running of the active system. When another alternate system is started, the synchronization process begins again.

Takeover

Starting the takeover

A takeover can be started by a number of events:

- The alternate system detects that the active system has signed off abnormally from the CAVM.
- The alternate system detects the disappearance of the active system's surveillance signal.
- The operator or an MRO-connected region that is taking over sends the alternate system a CEBT PERFORM TAKEOVER command.
- The operator issues a CEMT PERFORM SHUTDOWN TAKEOVER or IMMEDIATE command to the active system.

Whether there is a takeover after one of the above events, and the level of operator involvement in that takeover, depends on the event, and on the current system initialization table (SIT) takeover option. There are three SIT TAKEOVR options: AUTO, MANUAL, and COMMAND.

The active system signs off abnormally from the CAVM: If the active system signs off abnormally from the CAVM, for whatever reason, and the takeover option is **not** COMMAND, the alternate system will initiate a takeover.

The alternate system detects the disappearance of the surveillance signal: If the alternate system detects that the active system's surveillance signals have ceased, the action taken by the alternate system is dependent on its current takeover option.

If the takeover option is COMMAND, the alternate system will **not** initiate a takeover.

If the takeover option is AUTO, the alternate system will initiate a takeover automatically, as soon as the alternate-system delay interval (ADI) has elapsed.

If the takeover option is MANUAL, when the alternate-system delay interval (ADI) has elapsed, the alternate system sends a message asking the operator whether it should try to takeover, or ignore the apparent failure of the active system. If the operator can repair the active system, the alternate system can be told to ignore the loss of the surveillance signal. If the active system recovers, the alternate system will detect the reappearance of its surveillance signal, cancel the message to the operator, and continue with its standby role. If the operator cannot repair the active system, then the alternate system should be told to begin takeover processing.

A CEBT PERFORM TAKEOVER command is issued: This command may be issued to the alternate system by the operator, by another alternate system performing a takeover in a two-CEC MRO configuration, or by the overseer. On receipt of this command, the alternate system begins takeover processing, without reference to the operator, regardless of the takeover option.

A CEMT PERFORM SHUTDOWN TAKEOVER(IMMEDIATE) command is issued: This command can be used to initiate a takeover by instructing the active system to shut down, and sign off abnormally from the CAVM. However, a takeover will only occur if TAKEOVR = AUTO or MANUAL has been defined for the alternate system.

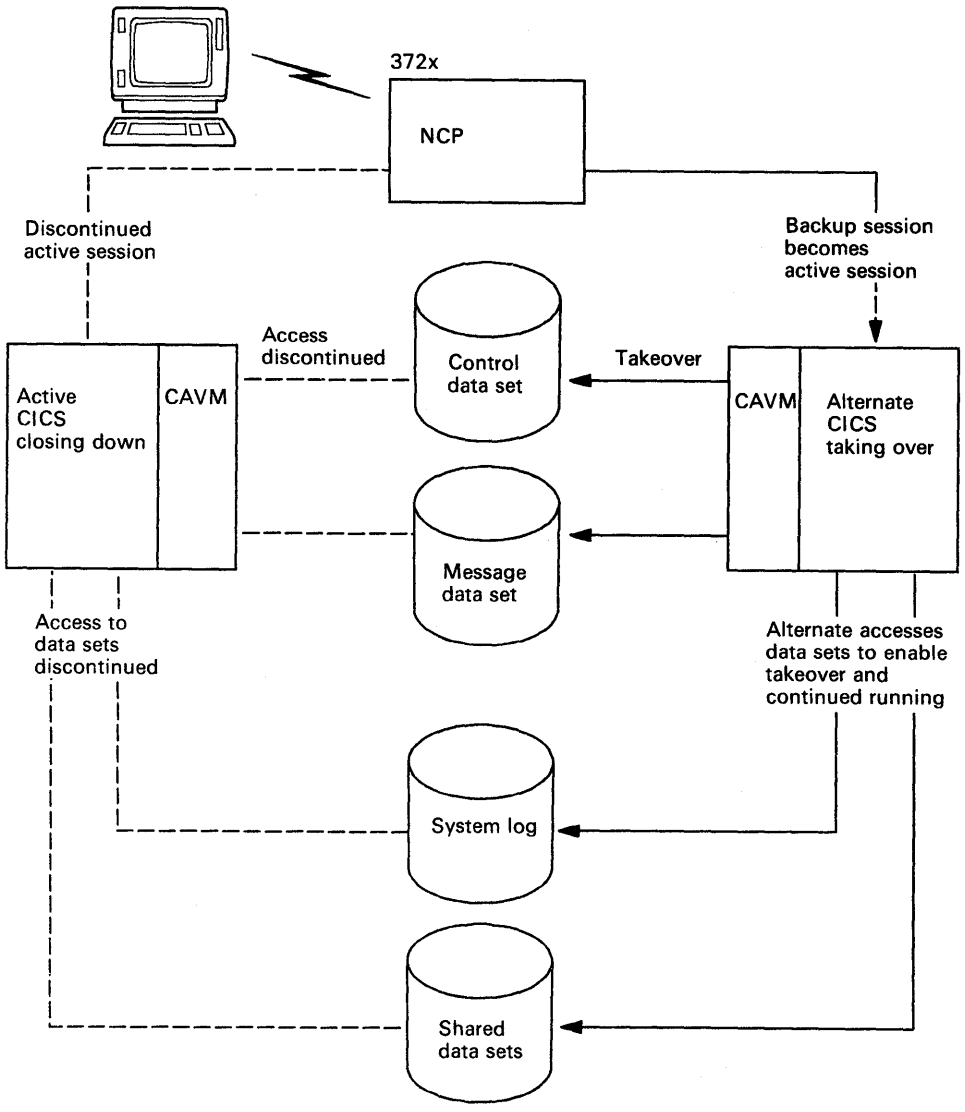


Figure 57. Takeover

Takeover processing begins

Once it has been decided that the alternate system will try to take over from the active system, a TAKEOVER request is passed to the CAVM. In most cases this request will be accepted, but may be rejected for any of the following reasons:

- The active system system has already signed off normally.
- The active system is not the same active system as the one that the alternate system had been tracking. The CAVM detects that it is a new active system, probably as a result of a restart in place. In this case, the alternate system cannot continue its role, and a new alternate system should be started.
- The active and alternate systems are on different CECs, and the alternate system has not been monitoring the active system's surveillance signals long enough to assess the difference between the time-of-day clocks.

When the CAVM has accepted the takeover request from the alternate system, an attempt by another CICS to sign on to the CAVM as an active system will be rejected. The alternate system next issues the MODIFY netname,USERVAR command to redefine the CICS application name,

and begins to switch the sessions of the XRF-capable terminals.

During takeover, the alternate system uses two different mechanisms to try to force the termination of the active CICS job:

- If the active system is still signed on to the CAVM, the alternate system uses the surveillance mechanism to try to pass a takeover-requested message to the active system, including a dump or no-dump indicator. If the active system receives the message, it responds by issuing abend U206, and eventually signs off abnormally from the CAVM.

- If the active job is still executing, the alternate system also issues a CANCEL command (prefixed by a JES routing command in a two-CEC configuration). The CANCEL is issued in case the active system is unable to respond to the alternate system's request to take over.

Next, the alternate system starts to process the command list table (CLT).

Checking for termination of the active system

The alternate system asks JES periodically about the status of the active system. If JES replies that the job has terminated, the next phase, ("Completing the takeover") can start immediately.

If JES replies that the job is still executing, the alternate system continues to ask about the status until the time interval defined by the JESDI SIT parameter expires. After that time interval, the alternate system asks the operator whether the CICS active job or the active system's CEC has failed. The alternate system will also ask this question if JES is not running, or does not respond.

Note that when active and alternate systems are running in different CECs, JES might continue to tell the alternate system that the active job is still running even though the active CEC or its operating system has failed. In this case, the alternate system cannot complete its takeover without operator intervention. Another possibility is that the active job is still running, and either never received the CANCEL command, or received it but cannot terminate because a system error, requiring a FORCE command, has occurred.

If the active system's CEC has not failed, the operator must ensure that the active job really has terminated before informing the alternate system that the active job has ended.

If the active system's CEC has failed, and the operator decides that an IPL is required, the operator should stop the processors of the failed CEC and then perform system reset. When the reset is successfully completed, the operator replies to the alternate system's question, telling it that the CEC has failed.

In this case, an internal record is kept that the CEC, identified by its SMF system identification (SID), has failed. Other alternate systems examine this record while they are performing a takeover, to try to avoid operator intervention.

The alternate system cannot complete takeover until the operator replies to its question, unless either:

- The alternate system receives a late reply from JES that the active job has terminated.

- A previous reply to another alternate system's message has already confirmed CEC failure.

If either of these events occurs, the operator does not have to reply, and takeover continues.

Completing the takeover

When CAVM has received confirmation that the active job has terminated, it notifies the alternate system that it may now assume the fully active role, and updates the CAVM control data set to this effect.

Emergency restart resumes, but, in a two-CEC environment, if the time-of-day clock of the new active system's CEC is slow compared with that of the old active system's, the takeover will be delayed until the new active system's time-of-day clock has reached the value of the old active system's clock at the time of job termination.

Then the alternate system completes its emergency restart, and becomes the active system. The restart time is improved by the existence of backup

sessions for XRF-capable terminals. The new active system does not have to establish new sessions for these terminals. It performs session cleanup for them, and re-establishes the sessions for other VTAM terminals. Terminal users still use the same generic applid/USERVAR to log on to CICS.

Logging and archiving

Because the aim is to provide a rapid recovery from a failure, your system log must be on two disk data sets. To avoid any archiving delay, and consequent unnecessary takeover delay, you should consider using the journaling replaceable modules to automate archiving. The *CICS/MVS Customization Guide* describes them.

If you submit the archiving job for execution on the active system's CEC, and that CEC fails while an archiving job is running, the job will have to be resubmitted, and takeover might be delayed until it finishes. This problem could be avoided by making a practice of submitting the archiving job for execution on the other CEC.

Failure analysis

Diagnosis information about the failure of the active system is provided by the usual termination dumps. Taking a dump is a part of the CICS job, and the alternate system cannot complete its takeover processing until the active job has taken its dump and terminated.

You are recommended to specify SDUMP as the termination dump, to provide adequate diagnostics, and to ensure that the active system closes down as quickly as possible. CICS provides an exit routine for the MVS PRDMP (printdump) service aid, to interpret and format the dump. For more information, see the *CICS/MVS Operations Guide*.

If the active system was running normally and it is being taken over because of a command from the operator or from another CICS region, no dump is taken.

After takeover

In a two-CEC environment, after the takeover, the operator performs a manual switch of any devices that need to be physically connected to the new active system, perhaps local VTAM, or TCAM terminals, or of other software outside the control of CICS.

After a terminal has switched to the new active system, your end user will have to sign on again, if sign-on security is in operation.

As in an emergency restart, an end user might have to reenter the last transaction, if that transaction was in-flight when the active system failed. This applies to all classes of terminals. If you prefer, you can send your own message, telling the end user what to do. You should consider your methods of establishing what was the last successful logical unit of work (LUW) before the takeover occurred, so that you can provide users with a meaningful recovery message.

Initiating network changes

To allow additional end users to log on after a takeover, VTAM must change the application name (specific applid) in its USERVAR table. The alternate system issues an MVS MODIFY vtamname,USERVAR command to change the entry in its local USERVAR table. Network operators at all other VTAMs that communicate with that CICS system must issue the MODIFY command to change the specific applid in their USERVAR tables. You can use the NCCF CLIST facility to automate this procedure. When the alternate system issues the MODIFY command, NCCF signals the change in applid to other NCCFs with which it is in session.

Reestablishing the system

When the old alternate system has become the new active system, there is a period when it runs without an alternate system as its partner. You should plan to start an alternate system as quickly as possible to restore the protection of XRF to your users. You can use the old active job's JCL for the new alternate job, ensuring that the correct START override is coded, or you can use different JCL. The job to start a new alternate system may begin execution when you know that the old

alternate system has become the new active system. This will probably be before the new active system has completely finished the takeover.

When the new alternate system has been initialized, new backup sessions are established for end users with XRF-capable terminals.

Elements of XRF

CICS, running in XRF mode, has the following functional elements, all of which will be described in more detail later:

Shared data sets
The XRF design is based on the use by the active and alternate systems of shared data sets. However, only two special data sets, the CAVM (CICS availability manager) data sets used for communication between the active and alternate systems, are accessed concurrently by the two systems. Others, such as the restart data set and system log, are accessed by the alternate system only after it has taken over from the active system. They are shared only in the sense that they must be physically accessible to the alternate system at the time of allocation, and must therefore be specified as DISP=SHR. User data sets that are to be accessed by the alternate system after takeover must also be shared in this way. Other data sets are unique to the active or alternate systems. For example, both active and alternate system have their own trace and dump data sets.

State management
Each active system is backed up by only one alternate system. If there is a number of CICS systems running in XRF mode, connected by multiregion operation, each active system has an alternate system providing XRF cover.

The active system is assumed to have ownership of, and authority to update, all shared data sets, apart from the special CAVM data sets. The alternate system cannot open these data sets until it has completed takeover, and has become the active system.

Surveillance
The active and alternate systems write and read regular surveillance signals to the CAVM data sets, and monitor each other's signed-on status.

Takeover
In a takeover, an alternate system removes its active partner, and assumes the role of the active system. If there are several active/alternate pairs working in an MRO (multiregion operation) two-CEC configuration, all the regions must be taken over. A master or coordinator alternate region instructs the other alternate regions to take over, based on the information in its command list table (CLT). The user can specify manual intervention in the takeover process.

Terminal control
XRF depends on VTAM's and NCP's ability to build a second, backup session from the communications controller to the alternate system, for eligible terminals. After a takeover, this session is switched, so that the terminals can communicate with the alternate system.

The alternate system is able to track the installations, logons, and logoffs of all terminals, and it attempts to reconnect these terminals after a takeover.

Message management
The active system sends messages to the alternate system about its terminals, so that the alternate system can keep its terminal tables up-to-date, and request backup sessions. Like surveillance, message transmission takes place through the CAVM data sets.

The following sections describe the structure of a CICS system running with XRF. You will find that, in some respects, the structure differs from that of a non-XRF system.

Task control blocks (TCBs)

When CICS executes without XRF, it uses one TCB for the main task, and other TCBs for specific pieces of work, for example, the VSAM/BSAM TCB is used to process file control, transient data, temporary storage, and journal control requests.

If you are using XRF, an extra TCB, the CAVM TCB, is used.

The CICS TCB and the CAVM TCB are loosely coupled in that there is some communication between them, for example, state changes, notification of events and messages. However much of the processing done by one TCB can be done independently of, and is irrelevant to, the other TCB.

Like the CICS TCB, the CAVM TCB uses extra TCBs to offload specific pieces of work, although for CAVM the reasons are functional rather than for performance.

Initialization

Signing on to the CAVM

Use of the XRF facilities is governed by the XRF and START SIT parameters. In all cases, the XRF CAVM interface modules (DFHXRP and DFHWCSLM) are loaded by the CICS nucleus build routine (DFHSIB1) and DFHXRP's static storage area is initialized. If XRF = NO is specified, no further XRF action is taken, except for the use of the system task to coordinate recovery tasks as described below.

If XRF = YES is specified, the CAVM is started by DFHSIC1. This is done by calling the **sign-on** service in DFHXRA, which in turn issues an MVS LINK to DFHWSSON. DFHWSSON attaches the CAVM TCB, which starts by performing numerous checks, in particular checking that the CAVM data sets are valid, and that the state is **valid**, that is, the system is not, for example, trying to sign on as an active system when there is already an active system running. DFHWSSON returns control when it has been determined whether the CAVM sign-on is to be accepted or not. If accepted, the CAVM TCB continues independently, performing its surveillance functions.

DFHSIC1 also examines the SIT START parameter. Its actions are the same as non-XRF systems, except for START = STANDBY. If STANDBY is specified, the opening of the CICS catalog is deferred, because the catalog is shared with the active system and cannot yet be accessed by the alternate system, and the SIT parameter is changed to EMERGENCY.

Other changes to the initialization process affect:

- DL/I
- Temporary storage
- Transient data
- Terminal control.

Eventually, control reaches DFHSII1, where a multisharing operational structure is used for resource recovery. The structures for XRF = NO, XRF = YES on the active system, and XRF = YES on the alternate system are different. The following sections illustrate the differences.

Task structure for initialization

A description of initialization when running with or without XRF is given in "Task Structure for Initialization" on page 154.

DL/I initialization

The sign on (or identify) takes place at a point following takeover (in DFHDLRP), to prevent two systems appearing to attempt sign on using the same name. To avoid losing the benefit of loading the DL/I code before takeover, DL/I initialization is split into two phases:

- DFHSIH1 loads the DL/I code.
- DFHDLRP performs the sign on, and checks if the user has specified DL/I data sharing.

Temporary Storage

DFHSIG1
DFHSIG1 determines the control interval (CI) size for the auxiliary temporary storage data set using RDJFCB and SHOWCAT macros, rather than the SHOWCB macro which requires the data set to have been opened.

DFHSIH1
DFHSIH1 allocates the major temporary storage control blocks: common area, auxiliary control area, buffer control area (BCA), VSWA control area (VCA), and so on.

DFHSIH1 also allocates the buffers for the auxiliary temporary storage data set.

DFHSII1
DFHSII1, executing under the III task, invokes DFHTSP to attach the temporary storage recovery task.

No distinction is made between the active system and the alternate system.

DFHTSRP
DFHTSRP allocates the initial temporary storage unit table (TSUT), and then initializes the auxiliary temporary storage data set. This involves:

- Formatting the (empty) data set on temporary storage cold start.
- Opening the data set.
- Reading the control record.
- Determining the current and high RBAs.
- Allocating storage for the temporary storage byte map.
- Restoring the temporary storage byte map and TSUTs on warm start. In a non-XRF system, this logic is in DFHSII1.
- Recovering the temporary storage byte map on emergency restart.

Transient Data

DFHSID1
DFHSID1 has two functions:

- To allocate the major transient data control blocks (TDST, MWCA, MBCA, and MRCA).
- To allocate I/O buffers for the intrapartition data set. The CI size is determined using RDJFCB and SHOWCAT macros rather than the SHOWCB macro (which requires the data set to have been opened). The assumption is made that the DCT contains intrapartition entries if the intrapartition data set is defined in the CICS job stream.

DFHSII1
DFHSII1, executing under the III task, invokes DFHTDP to attach the TD recovery task.

For alternate systems, this is done before DFHSII1 waits for takeover to be completed.

DFHTDRP
DFHTDRP creates the DCT index and then, for alternate systems:

- Dynamically creates control blocks for extrapartition destination CXRF. The address of the DCT entry is held in TDSTCXRF. The definitions of CXRF and DFHCXRF, in terms used to define entries in the DCT, are:

```
DFHDCT  TYPE=EXTRA
        ,DESTID=CXRF
        ,DSCNAME=DFHCXRF
        ,OPEN=INITIAL

DFHDCT  TYPE=SDSCI
        ,DSCNAME=DFHCXRF
        ,BLKSIZE=132
        ,RECSIZE=128
        ,RECFM=VARUNB
        ,BUFNO=2
```

  Note: Any attempt to code these entries in the DCT will be rejected.

- Opens destination CXRF.

DFHTDRP then builds the DCT index from the DCT.

For alternate systems, DFHTDRP waits for the restart data set (and, by implication, the other shared data sets) to become available.

DFHTDRP then:

- Restores/recovers the DCT.
- Opens those extrapartition data sets for which OPEN = INITIAL was specified.
- Posts TDSTECB1(TDSTDCTI) complete.
- Initializes the intrapartition data set if the DCT contains intrapartition entries. This involves:
  - Formatting the (empty) data set on transient data cold start.
  - Opening the data set.
  - Reading the control record.
  - Determining the current and high relative byte addresses (RBAs).
  - Allocating storage for the CI state map.
  - Recovering the CI state map on emergency restart.
  - Posting MRCAECB(MRCACSMI) complete.
- Posts TDSTECB2 (TDSTTDPI) complete.
- Recovers transient data AIDs on emergency restart.

DFHTDP

In general, transient data requests can be issued before transient data initialization is complete. However, they will be delayed. The extent of the delay will depend on the request and the destination type

- LOCATE and BROWSE are delayed until TDSTECB1 (TDSTDCTI) is posted complete.
- GETs and PUTs for extrapartition destinations are delayed until TDSTECB1 (TDSTDCTI) is posted complete.
- GETs and PUTs for intrapartition destinations are delayed until MRCAECB (MRCACSMI) is posted complete.

The exceptions are PUTs to CICS destinations, that is, those destinations whose DESTIDs begin with "C". Until TDSTECB1 (TDSTDCTI) is posted complete, DFHTDP treats such destinations as indirect, and TDSTCXRF is assumed to contain the address of the target destination.

CICS availability manager

The CICS availability manager (CAVM) is the component that provides the basic controls and services necessary to maintain the relationship between active and alternate systems.

This section describes:

- The CAVM interfaces with the rest of CICS
- The internal CAVM services
- The processes that support the interfaces.

CAVM interfaces with the rest of CICS

The CAVM has three main groups of interfaces with the rest of CICS:

- State services

  These are accessed using the interface module DFHXRA, and comprise:

  - Sign-on – join an active or an alternate system to an XRF pair.

- Sign-off – leave an XRF pair, either normally or abnormally. If the active system leaves the system abnormally, it invites a takeover by its alternate partner.
- Takeover (alternate system only) – takeover from the active system.

- Message services

  These are accessed using the interface module DFHWMS and comprise:

  - GETMSG (alternate system only) – retrieve a message sent by the active system.
  - PUTMSG (active system only) – send a message to the alternate system.

- Status exits

  The CAVM drives exits (established using sign-on) to convey status data:

  - NOTIFY – this exit (DFHXRB) is driven by the CAVM to inform the rest of CICS about significant changes in the status of its partner system.
  - Inquire status – this exit (DFHXRC) is driven by the CAVM to obtain status information for this CICS system.

The CAVM also has an interface with the CAVM in its partner CICS system. This interface is embodied in the protocols applying to the use of the CAVM data sets for communication between active and alternate systems.

CAVM runs under a separate TCB which is a subtask of the CICS TCB, and, apart from some of the interfacing code, runs above the 16-megabyte line. It is packaged as two load modules:

- DFHWSMS is resident throughout the time that the CICS system is signed on to CAVM, and contains the working set code.
- DFHWSSON contains most of the sign-on code, and is present only during sign-on.

The CAVM is structured as a set of long-running, cooperating processes created during sign-on, and running on top of a set of dispatching and LIFO storage services tailored to their needs. What the various processes do will be described later, but in the meantime, Figure 58 shows a list of the processes and their principal modules:

Process name	Main module	Process # (active sign-on)	Process # (alternate sign-on)
Clock	DFHWSTI	1	1
Takeover	DFHWSTKV	-	2
Primary status writer	DFHWSSW	2	3
Secondary status writer	"	3	4
Primary status reader	DFHWSSR	4	5
Secondary status reader	"	5	6
Message writer	DFHWMP1	6	7
Message reader	DFHWMG1	-	8

Figure 58. CAVM processes.

State services

The CAVM state services are used by a CICS system to modify its XRF status. State refers to the state of the active or the alternate system – **signed-on**, for example. Requests are issued by callers running under the CICS TCB passing the request by means of a parameter block described by DFHWSSDS.

Sign-on

CAVM sign-on establishes this CICS system as a partner in an XRF pair, and builds the CAVM surveillance structure. The SIGNON request is issued by the CICS initialization module DFHSIC1 using DFHXRA, which issues an MVS LINK to DFHWSSON, the transient CAVM module, entering it at DFHWSSNA in DFHWSSN1. The sequence of operations is then:

1. DFHWSSN1 issues an MVS ATTACH of DFHWSMS, and then waits for posting of the completion, or end-of-task event. The MVS task created by the attach is the CAVM TCB referred to above.

2. The CAVM TCB starts at entry point DFHWSMNA in DFHWSRTR. The parameters passed on the ATTACH indicate that it is a sign-on request (the same entry point is used by DFHXRA under the CICS TCB for sign-off and takeover requests).

3. DFHWSRTR calls DFHWSSN2 (and thence DFHWSSN3) to open and validate or initialize the CAVM data sets, check that the sign-on is valid, and update the state and status records to reflect the fact that a new system has signed on. DFHWSSN2 also establishes DFHWSSOF, the sign-off routine, as an ESTAE exit for the CAVM TCB.

4. DFHWSRTR then initiates the transition to XRF process mode by calling DFHWDINA to create the first process, passing it the address of DFHWSTI as its entry point.

5. DFHWDINA establishes the XRF process dispatcher, and starts executing the first process (at DFHWSTI).

6. DFHWSTI calls DFHWSXPI, which issues attaches for all the remaining XRF processes listed above, then posts the sign-on ECB that DFHWSSN1 is waiting for on behalf of the issuer of SIGNON. DFHWSTI then continues running as the clock process.

Sign-off

After a sign-off, the CICS system is no longer part of the XRF pair. Sign-off requests are issued by the CICS termination module DFHSTP, and by the recovery program DFHSRP. The calls are made using DFHXRA, which calls the CAVM module DFHWSMS (at entry point DFHWSSNA in DFHWSRTR). The sequence of operations is then:

1. DFHWSRTR (running under the CICS TCB) issues an MVS DETACH for the CAVM TCB.

2. The CAVM TCB abends as a result of the DETACH, and enters its ESTAE routine DFHWSSOF which updates the state and status records to reflect the fact that the system has signed off (either normally or abnormally, as specified by the caller of sign-off).

Note: DFHWSSOF is entered for abends of the CAVM TCB for reasons other than the SIGNOFF DETACH. Such failures also result in a sign-off.

Takeover

The takeover request is made to CAVM by an alternate system when it wishes to take over from its active partner. The request is issued by DFHXRCP (in response to a CEBT PERFORM TAKEOVER request). The calls are made using DFHXRA which calls the CAVM module DFHWSMS (at entry point DFHWSSNA in DFHWSRTR). The sequence of operations is then:

1. DFHWSRTR (running still under the CICS TCB) posts the ECB for which the CAVM takeover process is waiting (WCSTXECB), then waits for CAVM to post another ECB (WCSTCECB) indicating completion of the takeover, at least as far as CAVM is concerned.

2. The CAVM dispatcher recognizes that the takeover process can now be dispatched.

3. The takeover process (DFHWSTKV) controls the takeover sequence, communicating with other CAVM

processes where necessary, and updates the state and status records to reflect the fact that the system has become active. The details are more fully described under DFHWSTKV in "Chapter 3.2. CICS Executable Modules" on page 373.

4. The takeover process posts the ECB for which the issuer of the takeover request is waiting and then returns. The return terminates the process.

Message services

The CAVM message services are used by an active system to send resource status information to its associated alternate system.

The message service interface represents named FIFO message sequences. The message is transmitted by means of a wraparound sequence of records on the CAVM message data set. Fields in the status records in the CAVM control data set (WSASM and WSARM in DFHWSAPS) are used to communicate information about, for example, the point at which to start reading messages, and the possibility of overwriting unread messages.

The message services are invoked from a standard CICS transaction environment, by calling the message interface routine DFHWMS (its address is in the list pointed to by CSAXRFNT, set up by DFHSIB1), which passes a request parameter block as defined by DFHWMSPS.

The request is routed to the appropriate service routine using DFHWMS, DFHWMS10 and DFHWMS20. This sequence of routines makes the transformation from the standard CICS transaction environment (under the CICS TCB running below the 16-megabyte line with register 12 addressing a TCA, register 13 the CSA, CICS LIFO support, and so on) to the XRF-LIFO environment (with register 12 addressing a dummy XRF process block (XPB), register 13 a save area, and with XRF-LIFO support, though still running under the CICS TCB).

The routing is described by the control block structure set up by sign-on, in particular the link between the XRF static control block (DFHWCSPS) below the 16-megabyte line and the XRF global control block (DFHWCGPS) above the 16-megabyte line.

PUTMSG

An active system uses a PUTMSG request to send messages to an alternate system.

The message request router (DFHWMS20) calls the PUTMSG routine (DFHWMQP), then:

1. DFHWMQP places the request on the request service queue WMGPUTQ, and posts the message writer CAVM process. It then issues a KC WAIT macro for the request completion event.

2. The message writer process checks that there is an alternate system that can receive the message, copies the message into its output buffer, then posts the request complete.

3. DFHWMQP returns the response to the caller (using DFHWMS).

GETMSG

An alternate system uses the GETMSG request to retrieve a message sent by an active system for the specified message queue, and to place it in the data area provided.

If the queue is currently empty, the caller's CICS task waits until either a message arrives, EOD is signalled during takeover, or sign-off is issued.

After an alternate system has become an active system by issuing a takeover command, it can continue to use GETMSG to retrieve any queued messages which the old active system had sent before it finally terminated.

The message request router (DFHWMS20) calls the GETMSG routine (DFHWMQG), then:

1. DFHWMQG searches the input message queue data built by the CAVM reader process. If it finds a message in the requested queue, it passes the message back to the caller. If not, it waits for the reader process to post the queue, indicating that a message has arrived.

2. The message reader process continually reads input messages from the CAVM message data set and builds message queues in main storage. See the descriptions of CAVM processes on page 215 for more details.

Status exits

The CAVM drives exits to transmit status data to the rest of CICS. The exits are established at CAVM sign-on.

Notify

This exit (DFHXRB) is driven by the CAVM to inform the rest of CICS about significant changes in the status of its partner system. The exit runs under the CAVM TCB as part of whichever CAVM process happens to drive it. Most exits are driven by a status reader process when the status reader detects changes in the partner system's status record in the CAVM control or message data set. For more information, see the section on the status reader process on page 216.

The interface presented to the exit routine, and the events which cause it to be driven, are defined in DFHWNFDS.

The typical action for a notification is:

1. DFHXRB analyzes the event type, and creates a work element (DFHXRWPS), which it places on a work chain (XRSWECHN in the DFHXRP static storage (DFHXRSPS)).

2. The surveillance transaction (DFHXRSP), which was attached during CICS initialization by an INIT_SURVEILLANCE call to DFHXRA, takes work items from the chain and processes them.

Note: There is no response to the work items. DFHXRB does not wait to see what happens to them.

Inquire status

This exit (DFHXRC) is driven by the CAVM to solicit status information for this CICS system. The exit runs under the CAVM TCB as part of the clock process.

The status of the CICS system is communicated to its partner system as follows:

1. Status data is collected in the DFHXRP static area (DFHXRSPS) from other parts of CICS – for example, the status global user exit driven by TPEND.

2. When CAVM drives the inquire status exit, DFHXRC takes the status data from the DFHXRP static area, and passes it back to its caller.

3. The clock process copies the status data into the public status area (WSASIHR in DFHWSADS) for its own system.

4. A status writer process writes the status data, along with the rest of the status, when it next writes the system's status record to the CAVM control or message data set.

Internal CAVM services

The following sections describe the internal CAVM services:

- CAVM dispatcher
- CAVM LIFO storage
- CAVM trace
- CICS service routing.

CAVM dispatcher

This section describes the process structure which runs under the surveillance TCB.

The processes are created when the surveillance TCB is attached as part of sign-on processing, as described above. They perform a variety of functions, such as message management, and surveillance signal update.

The structure is similar to that of a CICS transactions running under the CICS TCB, and the processes are long-running and cooperative, unlike transactions which tend to be short and relatively independent.

Process management contains the following parts:

- Attach service – DFHWDATT

  This creates an XPB from a given initial entry point, initial data address, and ESPIE/ESTAE exit addresses.

- Dispatcher – DFHWDISP

  DFHWDISP runs the first ready process in its list.

- Initial attach service – DFHWDINA

  This is a service which performs dispatcher initialization, and establishes execution in the XRF process environment.

- Wait service – DFHWDWAT

  A process can issue a wait request specifying an MVS ECB and/or a set of internal events. At the same time, it specifies which locks it wishes to relinquish, and which new locks it needs before it is dispatched again. The dispatcher looks for some other ready process to dispatch.

- ESPIE/ESTAE – DFHWDSRP

  The ESPIE and ESTAE exits of the surveillance TCB will, among other things, run the ESPIE/ESTAE exits of the extant processes.

ATTACH

The CAVM attach service module is DFHWDATT. Its parameters are defined by DFHWDIPS in DFHWDSPS.

A new CAVM process is created and control returned to the issuer of the attach request. There is no dispatch cycle within attach. The new process is ready, and may be dispatched at any time after the attacher next issues a wait or detaches itself.

The new process is entered at the process entry point, with register 1 containing the specified initial data address.

The ESPIE routine is entered if a program interrupt occurs when the process is the one currently dispatched. It can elect to retry, or allow CAVM to terminate with message DFH6454, and the user abend 194.

The ESTAE routine is entered if an ABEND occurs. The ESTAE routine of each currently-attached process is called. There is no retry capability.

At first dispatch, a process is given the appearance of a standard MVS call to its initial entry (the corresponding return is treated as a detach – there is no explicit detach service). It holds no locks, and has a record of all broadcast events posted since it was attached.

Initial ATTACH service

The XRF TCB is attached, and receives control, as part of sign-on processing. The code which first receives control does not run as an XRF process, nor does it have XRF LIFO. Sign-on eventually issues an INITIAL_ATTACH request, and does not regain control unless the dispatcher returns, which it does only when it finds that there is no work to dispatch and no prospect of any, a situation which should not arise unless there is a logic error in CAVM.

The XRF process which is initially attached runs immediately in a standard XRF process environment. See the section about sign-on on page 210.

WAIT

The CAVM wait service module is DFHWDWAT. Its parameters are defined by DFHWDSPS.

The calling process is not redispatched until:

- If any events were specified, at least one of them has been posted.
- All required locks can be granted, because none is held by any other process.

The events which can be specified in a wait request are:

- External events

These are standard MVS events which are usually posted by some TCB other than the XRF TCB.

- Internal events

  These are like CICS events in that, though they resemble MVS events, they do not participate in the wait list. They are normally used for private wait/post communication between pairs of processes.

- Broadcast events

  Broadcast events allow a process to post events to be seen independently by all processes. There are 32 such events, represented by bit maps, each process having its own map. On a wait request, a process specifies those events that it wishes to broadcast, those of its own that it wishes to reset before waiting, and those for which it wishes to wait.

The locks can be specified from a set of 32 locks represented by a bit map. The locks are exclusive, and a process can hold any number of them. A process is initially dispatched holding no locks. On a wait request, a process specifies those locks that it wishes to release, and any additional locks that must be granted to it before it is dispatched again.

A hierarchy rule is enforced for locks, to avoid at least that source of deadlocks. Each lock has a level number assigned to it, and no process may request a lock if it already holds another lock with a level number that is no higher than that of the requested lock.

When the wait is issued, those events specified in events reset are marked as **not-posted** in this processes event record. Those events specified in events post are then broadcast. This means that they are entered into the events records of all processes, including the routine issuing the wait. The process then waits for any of the events in events wait, or either of the other events.

Also, when the wait is issued, those locks specified in lock release are relinquished, including any that may also be specified in locks required, and a check is made to ensure that locks required does not contravene the lock hierarchy rules.

When the process is redispatched, all events specified in events wait are set to **not-posted** in the processes events record, the process is granted those locks specified in locks required, and the following information is returned in the parameter block:

- The locks required mask contains a map representing all those locks currently held.
- The events wait mask contains a map representing the subset of those events originally awaited which were posted (whether already posted when the wait was issued, or posted subsequently).
- The events post mask contains a map representing all those events which are still marked as posted for this process.

CAVM LIFO Storage

CAVM has its own GET/FREE LIFO services which provide process-local LIFO storage above the 16-megabyte line. The entry points to the services, DFHWLGET and DFHWLFRE, are in the entry table addressed from the CAVM global control block (DFHWCGPS). The services can be invoked using the macros DFHWLG and DFHWLF.

The LIFO allocations are made from LIFO segments which form chains anchored in the owning process's XPB. Each segment starts with a header described by DFHWXLPS, and each allocation within a segment is preceded by a header described by WXLAHDR in DFHWXLPS.

The services are also used by code running under the CICS TCB. Such users, if currently running in a CICS environment (that is, with CICS LIFO) must be aware that the service assumes XRF LIFO, and builds a dummy XPB, usually in its own CICS LIFO, before calling. This dummy XPB serves only as an anchor for the LIFO mechanism, and is not known to the CAVM TCB dispatcher, or recovery mechanisms.

Both GET and FREE LIFO calls result in CAVM trace entries being made if the calls are made on behalf of a process running under the CAVM TCB.

CAVM trace

CAVM has a simple wraparound trace in main storage with the following characteristics:

- No external controls – it is always active and non-selective.
- Entries are of fixed size – 32 bytes.
- Entries are placed in a fixed-size trace area – 64K bytes.
- Entries are made only for XRF processes running under the XRF TCB.
- All entries contain two bytes of trace entry identification, the current process ID, the clock value (expressed using the same units and origin as the clock value in CICS trace entries), and 24 bytes of user information.

The trace entries and the interface to the trace routine DFHWTRP are defined in DFHWTRPS, and also described in a later section of this book. The trace area as a whole is defined by DFHWTGPS, and is addressed by WCGTRA in the XRF global area DFHWCGPS.

CICS service routing

Some CAVM code running as part of a CICS transaction under the CICS TCB runs in an XRF environment above the 16-megabyte line. In certain circumstances, it needs to invoke CICS services such as KC WAIT and SC GETMAIN, but cannot do so directly.

A routine in DFHWCSNT performs the necessary mode switching, using an extra register (register 15), and an XRF keyword is added to DFHKC and DFHSC to distinguish these cases.

CAVM processes

The following processes support the state services, message services, and the status exits.

DFHWSTI – clock

This process is the first to be attached (using the initial attach dispatcher service DFHWDINA). It starts by invoking DFHWSXPI to attach the rest of the CAVM processes, and posting WCSSOECB in the XRF static control block to indicate completion of sign-on. Its main purpose is to act as a clock service for the rest of CAVM. It performs this task by endlessly repeating the following steps:

1. Issue an STIMER macro for the clock interval (2 seconds).
2. Drive the CAVM inquire status exit to gather status data, and update the surveillance signal/status data in the public status record.
3. Post both primary and secondary status writer events (WSAGWEP and WSAGWES) to cause the status to be written out to a CAVM data set.
4. Issue an XRF wait to broadcast the tick event to all processes, and wait for the timer event.

DFHWSTKV – takeover

This process is attached only for an alternate system. Its primary function is to coordinate the takeover if this alternate system takes over from its active partner. It therefore spends a lot of time waiting for the posting of the takeover ECB WCSTXECB. It also validates the command list table (CLT), both initially and whenever a sign-on event is broadcast by the status readers. The CLT defines the hierarchy of regions in an XRF environment. This allows it to check, without waiting until takeover, that the CLT exists, and that it authorizes canceling the active CICS job.

If and when the takeover ECB is finally posted, this process controls the takeover sequence, and then posts the takeover complete ECB (WCSTCECB). Finally it returns, causing the process to terminate, since it is no longer needed now the system is an active one.

DFHWSSW – status writers

CICS with XRF has two status records, one in the CAVM control data set, the other in the CAVM message data set. There are two status writer processes, one for each status record. The two writers communicate using a shared control block (SRVCOM in DFHWSRDS) addressed by WSADSRCP in DFHWSADS and allocated by DFHWSSN2 during sign-on.

DFHWSSR – status readers

There are two data sets containing status records, the control data set and the message data set. There are two reader processes, one for each data set, which read the status records of partner systems from those data sets, and build public status areas in main storage which contains the most up-to-date status, and are available for inspection by other processes. The two readers communicate using a shared control block (SRVCOM in DFHWSRDS), addressed by WSADSRCP in DFHWSADS, and allocated by DFHWSSN2 during sign-on.

DFHWMP1 – message writer

This process serves the PUTMSG work queue and places message records on the CAVM message data set.

DFHWMP1 is first invoked by DFHWMI, which is in turn called by DFHSXPI, during sign-on. It initializes the message writer by allocating a work queue anchor (WMGPUT in DFHWMGPS), and attaching itself as the writer process.

When dispatched, the process waits until PUTMSG processing can begin. This is signaled by the posting of WMGPMECB, which, for a system which started as an alternate system, is not done until a takeover has been requested and has progressed to the point where the final active status is known.

DFHWMP1 then calls DFHWMWR to build output buffers and control blocks, and enters an endless request-processing loop.

The processing loop consists of:

1. An XRF wait on the ECB associated with the PUTMSG work queue and an alternating set of calls to DFHWMQS.
2. A sequence of alternating calls to DFHWMQS and DFHWMWR to select work queue elements and place message records in the output buffer. This continues until the work queue is exhausted.
3. A call to DFHWMWR to force the output buffer, if there are unwritten messages which specified FORCE.

In general, the output buffer is not written until either a new one is needed to accommodate the next message, or the last step is reached. Messages without FORCE can wait for some time in the output buffer before they are finally written, and made visible to an alternate system.

To be made visible to an alternate system, the message records must be written to the CAVM message data set, and the alternate system must be told they are there. This is done by maintaining an active write cursor (AWC) in the active system's status record (WSASMAWC in DFHWSAPS). The AWC indicates the position of the end of the last message record in the message data set, and consists of a cycle number and relative byte number (RBA). The cycle number is one more than the number of times the message data set has wrapped since the data set was initialized, and the RBA is the standard VSAM RBA. The AWC is updated in the active's public status every time a message CI has been successfully written.

When DFHWMWR is called to write a message record into its buffer, it examines the public status of the alternate system (provided by the status reader processes). If an alternate system is signed on, but this is the first time the message writer has seen it, DFHWMWR writes an initial read cursor (WSARMBRC in DFHWSAPS) equal to the current AWC into its own public status. The alternate system, when it sees this initial read cursor, starts reading messages from that point, and maintains a backup read cursor BRC (in its own copy of WSARMBRC) reflecting its progress in reading the messages.

DFHWMWR uses the BRC value to determine whether there is a risk of overwriting messages which the alternate system has not yet read (when the alternate system is lagging by a whole cycle).

The message writer process has an XRF ESTAE exit (another function of DFHWMP1). If driven, it calls DFHWMQS to terminate the process. Termination consists of posting any outstanding PUTMSG requests, and closing the message service, so that subsequent PUTMSG requests are given an immediate failure response.

The message writer process also has an XRF ESPIE exit (another function of DFHWMP1). If driven, it calls DFHWMPG to determine whether the program check occurred while moving data from user storage to the output buffer. If this is the case, it attempts to recover by terminating the current PUTMSG request.

DFHWMG1 – message reader

This process reads messages from the message data set, determines which are relevant to this alternate system, and creates input queues according to the queue names in the messages.

DFHWMG1 is first invoked by DFHWMI during sign-on to initialize the message reader. If the sign-on is for an active system, there is no occasion to read messages, and the initialization consists of arranging that an end-of-data response be given to any GETMSG requests.

For an alternate system sign-on, DFHWMG1 attaches itself as the reader process.

When dispatched, the process first calls DFHWMRD to initialize. DFHWMRD waits for the appearance of an initial read cursor in the active system's status (see the previous section about the message writer). It waits for either of two broadcast events, status change from a status reader, or final active status available from the takeover process.

Once the initial read cursor is seen, it builds input buffers and control blocks. DFHWMG1 then enters a message reading loop which:

1. Calls DFHWMRD to read the next message record. If the BRC has caught up with the AWC, DFHWMRD checks that the system is still an alternate one. If it is, it waits for the AWC to move, then reads the next record. If the system is now an active one, it indicates end-of-data.
2. Calls DFHWMQH to place a copy of the input message on the appropriate queue in main storage, from which it will be retrieved by DFHWMQG in response to a GETMSG request.

   If DFHWMRD has indicated end of data, DFHWMQH is called instead to clean up, and prepare to terminate, the message reader process.

The message reader process has an XRF ESTAE exit (another function of DFHWMG1). If driven, it too calls DFHWMQH to terminate the process. Termination consists of posting any outstanding GETMSG requests, and then closing the message service, so that subsequent GETMSG requests will be given an immediate failure response.

CICS availability manager SVC services

This section describes the CAVM extensions to SVC services, used principally by CAVM state management.

CAVM overview

This section is mainly concerned with the takeover initiation service. For an alternate system to perform a takeover, CAVM state management must ensure that the active job has ended. Also, if the active system is one of a set of MRO-connected systems, and the alternate system is on a different CEC from the active system, the commands in the CLT must be issued to make the other MRO-connected systems perform takeovers.

Certain other CAVM SVC services, which form part of takeover initiation, are also separately requested by CAVM state management. The verify CLT service is an example of this. For a takeover initiation request, the contents of the CLT have to be checked to ensure they can be used by a particular alternate system. For example, the specific APPLID of the alternate system should be specified in the CLT. So that the appropriate consistency checks are performed in advance of a takeover, state management issues a request for the verify CLT service when the alternate system issues SIGNON to the CAVM.

Each CAVM SVC service is implemented as a separate request type to the CAVM takeover initiation program, DFHWTI. To invoke a service, the requesting module builds an argument block containing a CAVM service request type and input parameters as appropriate. It then issues the CICS SVC to invoke DFHCSVC, specifying the XRF DFHCSVC request type. On detecting the XRF request type, DFHCSVC in turn passes control to DFHWTI. On return from DFHWTI, a return code is supplied in register 15, together with the original CAVM service request type and reason code in register 0. Source member DFHWTADS contains specifications of the DFHWTI request types and input parameters.

The request types for CAVM SVC services are:

- Takeover initiation
- Job status inquiry
- CLT verification
- Overseer operator command
- CLT processing
- MODIFY USERVAR command.

For the overseer request see "The XRF overseer" on page 241. For the MODIFY USERVAR request see "MODIFY USERVAR" on page 239. For information about the way the other CAVM SVC services are used, see "CICS availability manager" on page 208.

DFHWTI is composed of a number of routines. Each routine has its own local storage and register save area, in addition to the module's global working storage area. Each DFHWTI request type has an associated routine which provides the necessary function. For example, the function for the takeover initiation request type is implemented by routine TIPENTRY.

Takeover initiation (TIPENTRY)

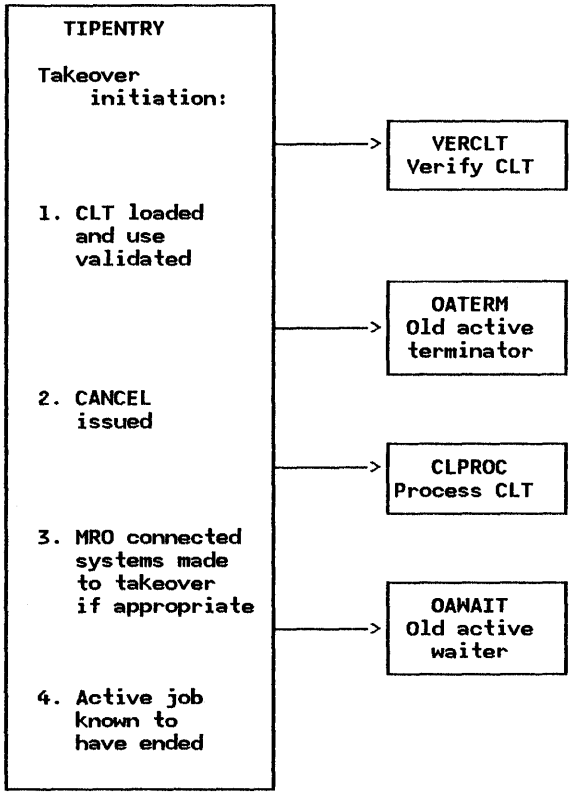


Figure 59. Takeover initiation routines

Figure 59 illustrates the calls to internal routines made by TIPENTRY and the functions associated.

The TIPENTRY routine initially calls routine VERCLT to load the CLT specified for use by this alternate system and to check its contents. One of the checks is to verify that the CLT contains the specific APPLID of this alternate system. If it does not, this alternate system is not authorized to issue CANCEL for the active CICS job. It is also not possible to determine which CLT commands have to be issued. If this authorization check fails, the condition is recorded, but the takeover initiation is continued.

If the active system is on a different CEC from the alternate system, MVS/XA SYSEVENT WKLDCHG is issued to request preference from the MVS/XA System Resources Manager during the current period of takeover.

If the authorization check made in the call to VERCLT was successful, TIPENTRY issues the CANCEL for the active CICS job by calling the old active terminator routine, OATERM. This routine constructs the appropriate system operator command to cancel the active CICS job, and issues the command under program control. Also the time-of-day clock value is recorded when the CANCEL has been issued.

TIPENTRY then calls the command list processor routine (CLPROC) requesting that the list of commands specified in the CLT for this alternate system is issued. TIPENTRY also passes to CLPROC an indication of the success or failure of the authorization check made in the earlier call to VERCLT.

Termination of the active CICS job is detected by TIPENTRY which calls the old active waiter routine, OAWAIT. This routine uses the inquire job status service provided by routine CHECKT (check termination). CHECKT detects termination of the active CICS job from JES or from the internal record of failed CECs, the CEC dead data. It also requests operator intervention if the period specified by the JESDI initialization option expires after the CANCEL has been issued by OATERM. Operator intervention is also requested if the authorization check made by VERCLT failed.

Issuing the CANCEL command (OATERM)

The old active terminator routine, OATERM, initially calls the check for termination routine, CHECKT, to verify that the active system is currently executing. If the check is valid, the CANCEL command is then constructed.

The syntax of the CANCEL command to be constructed depends on whether the active system is on the same CEC as the alternate system, or on a different CEC. An input parameter for the takeover initiation request specifies this. For the different CEC case, an operate CLT function makes a security check, and supplies the appropriate JES command routing prefix. The security check is to verify that the job name to be used on the CANCEL is specified in the CLT. This particular operate CLT function is implemented by an internally callable routine OPCLT with request type VALIDATE-TERMINATE.

The CANCEL command syntax used is:

```
JES-command-routing-prefix
  CANCEL jjjjjjjj,A=aaaa

where jjjjjjjj = jobname of
                 active CICS
      aaaa     = address space
                 identifier
```

See "Accessing the CLT (OPCLT)" on page 221 for a description of the JES command routing prefix.

After it has constructed the appropriate CANCEL command, OATERM issues the command using SVC 34 (MGCR), and then records the time-of-day clock.

Waiting for the active system to terminate (OAWAIT)

The process in routine OAWAIT to detect termination of the active CICS job can be viewed as having two stages.

The first stage occurs if the CANCEL command

was successfully issued by OATERM. The inquire job status function provided by routine CHECKT is repeatedly called with a short wait between calls. This continues for the period specified by the JESDI initialization option, or until termination is detected.

The second stage occurs if operator intervention is required. Examples of this are:

- OATERM could not issue the CANCEL, as a result of an error in the contents of the CLT.
- The JESDI period expired before termination of the active CICS job was detected.

Messages DFH6561, or DFH6562, are issued to request the system operator to ensure termination of the active CICS job. If OAWAIT is called with an indication that the authorization check failed, messages DFH6577 or DFH6578 are issued instead. While the operator reply is outstanding, the process of repeating the inquire job status requests is continued. If termination is detected, message DFH6563 or DFH6564 is issued to the system operator to inform him, and the MVS/XA DOM macro is used to delete the outstanding operator reply. Alternatively, if the operator confirms termination, this completes processing by OAWAIT. However, if the operator replies that the CEC of the active CICS job has failed, OAWAIT issues a request to update the internal record of failed CECs, the CEC dead data. This is done by OAWAIT calling the CEC dead data accessing routine, OPCDATA, with a PUT request. This creates a CEC dead data element with the MVS SMF identifier of the active CICS job's CEC and the time-of-day clock value recorded immediately after the WTOR to request operator intervention.

Validating use of the CLT (VERCLT)

Routine VERCLT provides the verify CLT service. This service is invoked internally to DFHWTI and externally. Examples of such calls are the internal call made for the takeover initiation request type, and the external call made by DFHWSSN2 as part of CAVM sign-on.

VERCLT obtains the specific APPLID and the CLT load module name from the AFCS. The routine which provides the operate on CLT service, OPCLT, is called, requesting that the CLT verification checks are made. This is the OPCLT request type CHECK.

For externally called requests, there are two additional steps. OPCLT is called again, specifying request type VALIDATE-TERMINATE to check the CLT has the necessary contents to issue the CANCEL command. The CLT is then deleted.

Issuing the commands in the CLT (CLPENTRY amd CLPROC)

A process CLT function issues the commands and WTOs in the CLT. Like the verify CLT function, this function can also be invoked internally and externally. However, in this case the external call is implemented by a process CLT external entry routine, CLPENTRY, and the internal version by routine CLPROC.

CLPENTRY calls VERCLT to verify use of the CLT, then calls the internal version, CLPROC, to issue the commands and WTOs. It then deletes the CLT.

CLPROC issues the commands and WTOs specified in the CLT for this alternate system by a process of repeatedly calling the operate CLT routine, OPCLT, and issuing the SVC 34 (MGCR), or WTO calls as appropriate. The OPCLT request type GETNEXT is used to locate successive command and WTO entries in the CLT.

If CLPROC is called with an indication that the authorization check made in VERCLT failed, CLPROC does not perform this process to issue the commands and WTOs, but instead issues message DFH6576.

CLT structure

The CLT is constructed by the DFHCLT macro using three CSECTS. The precise format of the CLT is defined with the macro call DFHCLT TYPE=DSECT. The overall structure is illustrated in Figure 60 on page 221.

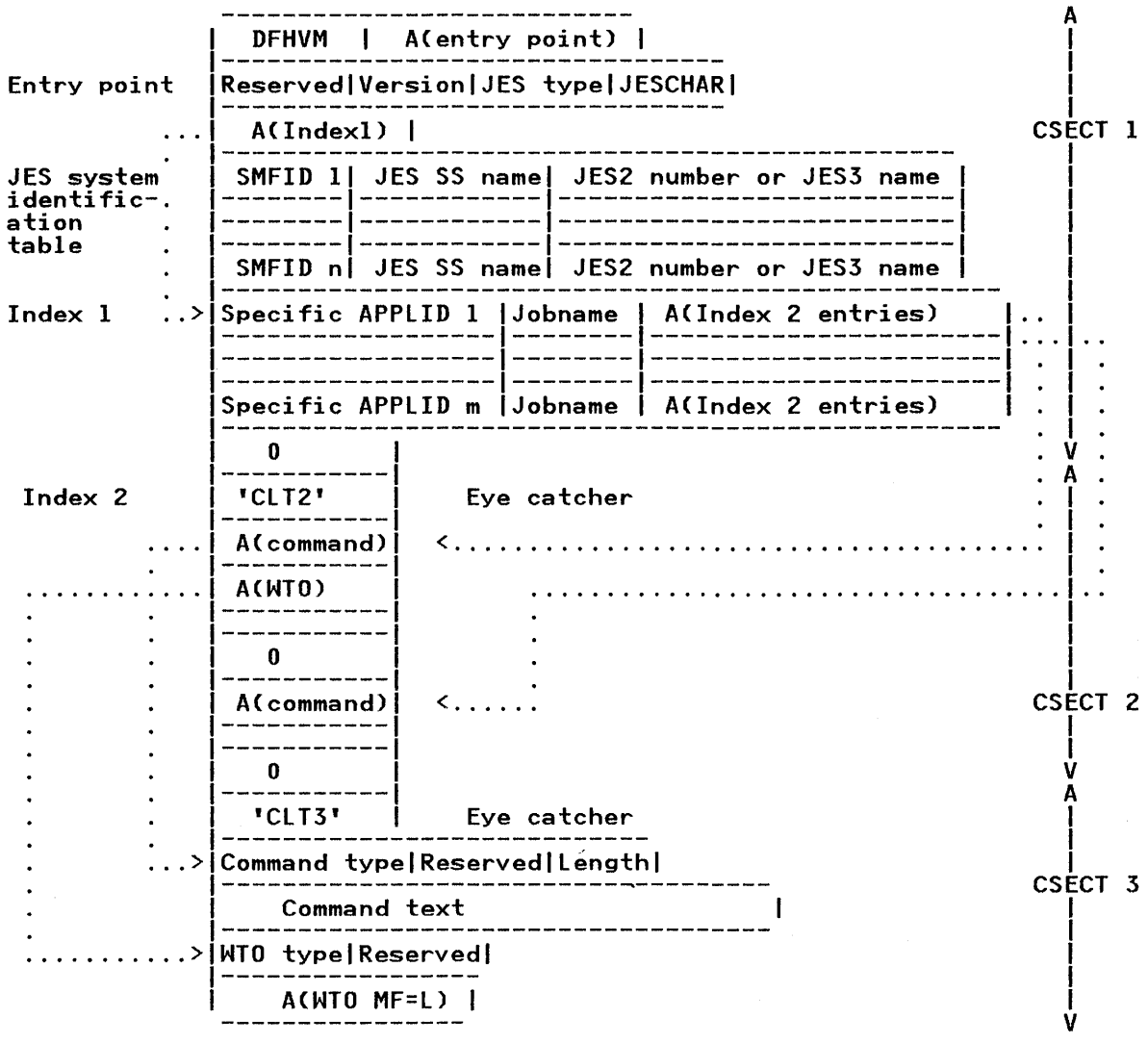


Figure 60. Structure of the CLT

Accessing the CLT (OPCLT)

After the CLT is loaded, it is accessed by the operate CLT routine OPCLT. This has the three subfunctions CHECK, VALIDATE-TERMINATE and GETNEXT:

- CHECK verifies that the CLT is a valid CLT load module. It also checks that it contains the specific APPLID of this alternate system. This is the authorization check for takeover initiation, and for process CLT. Messages DFH6565, DFH6566 and DFH6573 can be issued if checks fail.

- VALIDATE-TERMINATE checks that the job name to be used for a CANCEL is in CLT Index 1. If the active system is on a different CEC, the JES command routing prefix is then constructed. This involves a search of the CLT JES subsystem identification table. The syntax of the prefix is:

  For JES2: **cMn;**
  for JES3: **cSEND, mmmmmmm,**

  where:
  c is the JESCHAR
  n is the JES2 multi-access spool member number
  mmmmmmm is the JES3 MAINPROC name

  Messages DFH6568 to DFH6570 can be displayed if errors are found by

VALIDATE-TERMINATE in the CLT contents.

- GETNEXT, on its first call, locates the entry in CLT index 1 for the alternate system, thereby locating the start of a set of CLT index 2 entries. The CLT index 2 entries indicate the command and WTOs to be issued by this alternate system. Successive GETNEXT calls select the address of the next command, or WTO entries, from CLT index 2 until the end of the set of entries is reached.

Determining if a job is running with inquire job status (CHECKT)

The routine CHECKT (check termination) is the implementation of the inquire job status request type. Like certain other routines in DFHWTI, CHECKT is invoked by internal and external calls.

Two methods are used to find out if a job is running.

If the active CICS job is running on a different CEC, routine OPCDATA (operate on CEC dead data) is called, with an INQUIRE request to determine if there is a CEC dead data element for the active CICS job's CEC, that indicates it has failed. If so, the active CICS job is assumed to have ended, and control is returned with a code indicating the end of the CICS job.

If the active CICS job is on the same CEC, or there is no indication of CEC failure in the 2-CEC environment, CHECKT calls routine INQJES (inquire JES) to determine whether or not JES has a record that the active CICS job is running. The outcome of this request is then returned to the caller.

Accessing the record of failed CECs (OPCDATA)

The record of failed CECs, the CEC dead data, is a set of chained data elements in storage which is not freed, and which are anchored to the subsystem anchor block (SAB). The SAB is in turn anchored to the subsystem communications vector table (SSVT) for CICS. The SAB and SSVT exist across invocations of CICS. Each CEC dead data element records the MVS SMF identifier of a CEC, and the time-of-day clock value when a CEC is known to have failed. OPCDATA provides an INQUIRE and a PUT function.

For both INQUIRE and PUT, OPCDATA issues a request using MVS SSI VERIFY to obtain the address of the SSCT. It then serializes with all other CEC dead data accesses by obtaining the MVS local and CMS locks. The locks are released after the INQUIRE or PUT requests are processed.

For INQUIRE, OPCDATA searches the CEC dead data elements for an element containing the MVS SMF identifier of the active CICS job's CEC. If there is one that contains a time-of-day clock value later than the time of day supplied as an input parameter (time-of-day clock value at CAVM sign-on of the active system), the active CICS job is assumed to have ended.

For a PUT request, OPCDATA attempts to search for a CEC dead data element containing the MVS SMF identifier of the active CICS job's CEC. If found, the time-of-day clock value supplied as an input parameter is stored in the element. If not found, the necessary storage is obtained, and a newly created element is enchained as appropriate. If, however, there is no SAB, this is also created and anchored to the SSCT. A compare and swap instruction is used to anchor the SAB, to allow for its concurrent access by DFHIRP.

Determining from JES whether a job is running (INQJES)

Routine INQJES issues a STATUS request for the active CICS job to JES using the MVS SSI. To allow for this request being held up by JES, the actual issuing of the MVS SSI macro request, IEFSSREQ, is performed under a TCB attached for the purpose (routine IJESSUB). If the subtask does not respond within an acceptable time, the main routine, INQJES, returns control to its caller with a return code reporting the condition. If INQJES is entered again later, and is able to determine from subtask related storage that a previously outstanding request is complete, the outcome of this request to JES is returned to the caller.

XRF terminal control

This section consists of:

- "What tracking does"
- "XRF tracking streams"
- "Standby BIND processing"
- "Terminal Switching"
- "Reconnect processing"
- "MODIFY USERVAR."

What tracking does

A significant cause of delay during emergency restart in a CICS system without XRF support is the time taken to BIND the terminals in the network. This is caused by the path length in VTAM, and network delays due to I/O and path length in the network control program (NCP) and the logical unit (LU) microcode.

Standby BINDs

SNA architecture has been extended to reduce the delay caused by the path length in VTAM and NCP. It prepares that part of the path from the LU to the application which lies between the application and the boundary function in the NCP.

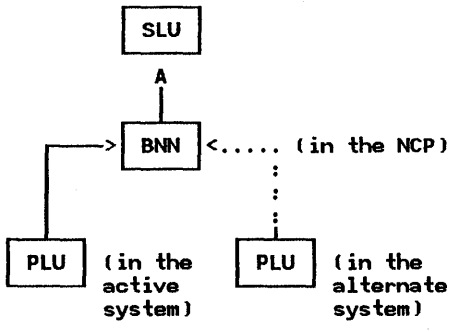


Figure 61. The standby session

It is known as a standby session, and can be converted into a real session by any of a set of SWITCH DATA TRAFFIC messages on either of the two sessions. It is established when the alternate system sends a form of BIND request, called a standby BIND, to the LU. The standby BIND flows no further than the NCP boundary function for the session, which checks that the session parameters in the standby BIND are compatible with the session parameters in effect for the active session. In particular, a security key – the correlation ID – which is carried in a control vector after the BIND proper, must match that supplied in the active BIND. The boundary function returns a response, either negative or positive, on the new backup session.

If the alternate system takes over, the backup session is converted to an active session by means of a SWITCH request from the alternate system. This causes the active session to fail, if it has not failed already. The positive response to SWITCH, returned by the boundary function, contains an extended response RU. This gives information about the states of the session at the time of the switch.

With XRF, terminal types fall into different classes of service. For more information, see the CICS/MVS XRF Guide.

Why we need to track instead of just reading the catalog

The active system must notify the alternate system whenever an active session that is to have a backup session is established. This notification is passed using the tracking mechanism. The same occurs when a session is ended.

Standby BINDs are issued using the usual DFHZCP process (DFHZSIM, DFHZLGX, DFHZOPN, DFHZOPX and DFHZNAC), and a terminal definition (TCTTE) is required to do this. During normal emergency restart, terminal definitions are read from the catalog. The catalog is a portion of the restart data set, and is described in the *CICS/MVS Recovery and Restart Guide.* However, the catalog is not available during tracking, so information equivalent to the catalog record must be sent from the active system to the alternate system by some other means. Again, this information is passed using the tracking mechanism.

Note: The catalog becomes available during takeover, and can be used as a source of supplementary information even if the resource manager does tracking.

Generalized catalog

When the active system adds or deletes a catalog record, it requests the alternate system to do the same. The alternate system may be seen as a re-implementation of the CICS catalog. The records in the CICS catalog form a structure, with the keys giving the position of each record in the whole.

Tracking a structure

XRF tracking enables the alternate system to maintain an image of the structure in the active system. To make this possible, the active system sends add and delete requests for nodes in the structure. Add requests carry data for the new node. For example, the session TCTTEs for LU6.2 are regarded as dependent upon the mode entries, which are in turn dependent upon the system entries. The records to add and delete TCT entries are sent as TCT contents messages. The session state (when bound) is regarded as an extra node beyond that for the TCTTE.

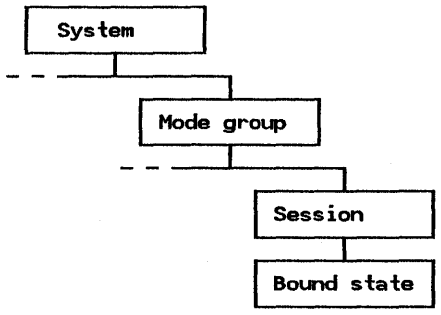


Figure 62. Advanced program to program communication resources as a structure

Data and key

For an add, the key places the data in the structure. For a delete, the key identifies the node to be deleted. If the node has dependent nodes, you cannot delete it.

The queue organizer – overview

Some adds, such as adding the session bound node to a TCTTE, and some deletes might take an appreciable time to complete. Meanwhile, adds and deletes for unrelated nodes can be processed. However, further adds and deletes for the node in question, and its dependents, must be held up until the first operation is completed.

This is managed by the tracking queue organizer, which saves all the actions in a data structure accessed by tracking key, and dispatches work at the head of the queue for each node. The resource manager can return:

done
The queue organizer carries on with the next action for the node.

scheduled
The queue organizer dispatches no more work for the node until the resource manager makes a subsequent **post** request, quoting a token passed on the original call by the queue organizer.

Terminal switching process

When the alternate system is requested to take over, the TCT is searched for all the TCTTEs that were bound with an XRF backup session. The SESSIONC CONTROL=SWITCH macro is issued for these terminals. This macro switches the terminal's session from the active system to the alternate system. The response data returned to this command details the last flows on this session. This data is used to determine what operations must be taken to clean up the session.

Session clean up: A terminal may have been in mid-transaction at the time it was switched. That transaction will have been backed out in the normal emergency restart manner, but the session remains in a busy state. The session now has to be cleaned up, to allow the end user to continue operation. The alternate system examines the SWITCH command response data to determine the state the session was left in when it was switched. It then initiates the necessary action to clean up the session to restore it to a normal not-busy state.

Session recovery: You can select various forms of recovery notification to inform the user that a takeover has taken place. After the alternate system has completed its initialization, this notification takes place.

For more information about terminal switching, see "Terminal Switching" on page 233.

The reconnection process

CICS attempts to acquire terminals defined as AUTOCONNECT(YES) shortly after control is given to CICS.

Autoconnect retries: If you are not using XRF and the attempt fails, there is no retry, and the terminal is not bound. After an XRF takeover, there will be a period in which terminals are physically switched to the new active system, and the network is reconfigured by means of VARY commands. Autoconnect is more likely to fail for temporary reasons. So, after takeover, the reconnect process is run and re-run at increasing intervals, for about half-an-hour, or until all the terminals that are marked to be reconnected are bound.

ZCP workload spreading: The first priority after an XRF takeover is to get the XRF-capable terminals (those terminals with backup sessions) back to work. To avoid interfering with this, the first autoconnect attempt is delayed by up to four minutes, based on the number of standby sessions that needed switching. The user can further extend this delay using the SIT keyword, AUTCONN. Don't confuse the AUTCONN keyword with the AUTOCONNECT option for defining terminals.

Warm starting session-states: Since the state of sessions on the active system is being tracked, it is possible, in effect, to warm start even the session-state of VTAM terminals without backup sessions. The AUTOCONNECT setting for a terminal is regarded as the cold start session-state, and is honored:

- If no tracking is done (that is, at a CICS cold start).
- For warm and emergency restart.
- For XRF takeover for terminals which are defined not to have their session-state tracked.

When the alternate system gets a session-state tracking message for a terminal, it takes precedence over the AUTOCONNECT setting.

There is more information about the reconnection process in "Reconnect processing" on page 237.

XRF tracking streams

Messages are sent by the active system in complete batches called streams. The start-of-stream message identifies a number of types of message in the stream. All the messages in a stream are assumed to be ordered in time. The individual types are ended one at a time. When the last type is ended, the stream is considered to have ended.

All messages carry an identifier. All messages in a particular stream carry the same identifier.

XRF tracking transmits updates on a special-purpose stream, the broadcast stream, which uses a reserved value of identifier (zero), and sends catch-up information using specific streams with non-broadcast IDs.

Each message also carries a key. This gives the position of the message data in the structure.

An interface macro, ZXTR, is defined in copy file DFHZXTR. This encapsulates the code used to send and receive messages in the active and alternate systems. This code calls the XRF message manager to transmit the records.

Figure 63 on page 226 through Figure 66 on page 228 illustrate the description in the text. Keywords are printed in upper case.

Preparing to receive a complete stream

See Figure 63 on page 226. The receiver issues initialize and open for input parameters, using the **nextstream** option to accept messages for the next stream that starts up. Routine1 gets control when this happens, as described below.

The value "B" in the example's ID parameter correlates the **open** to any other requests specifically for this stream, elsewhere in the same module.

The loop containing **run** serves all the streams, and runs until the message manager indicates end-of-data, or an error occurs. When an error does occur, there is no need for error recovery, because streams can be re-sent in their entirety.

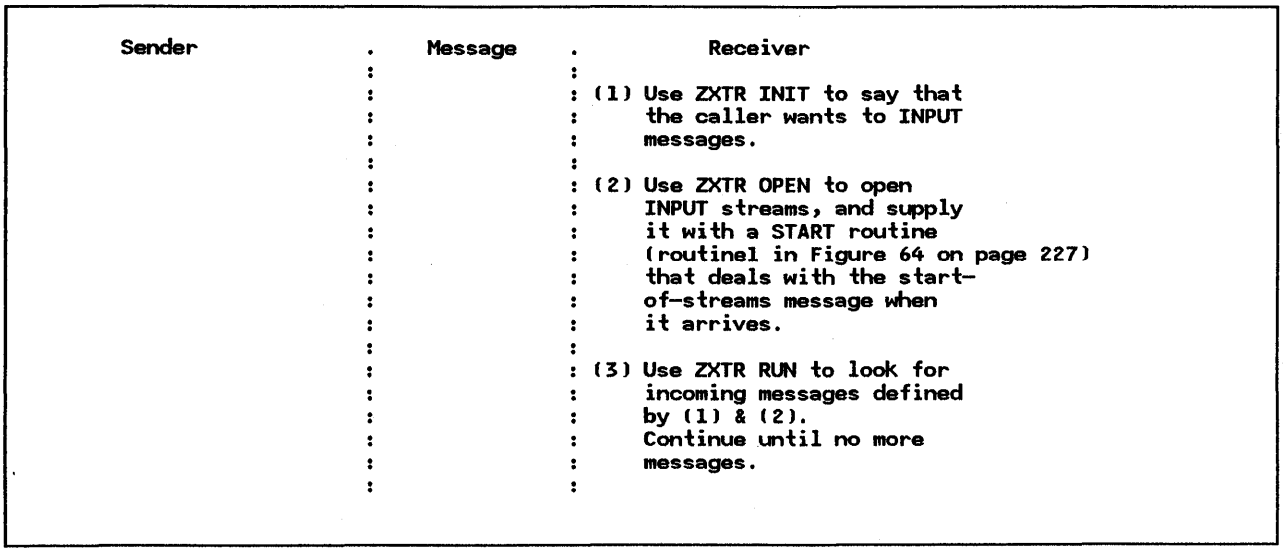


Figure 63. Tracking streams – an example (part 1 of 4)

Start-of-stream

The sender starts a fresh stream using the **open** parameter, as shown in Figure 64 on page 227. The **cold** option declares that this stream is not the continuation of a previous stream, and will be complete in itself. **Nextstream** causes a new stream-ID to be assigned from a global counter.

When the start-of-stream message reaches the receiver, the routine named in the **start** keyword of **open** is given control. This routine:

1. Notes the list of types, so that end-of-stream can be detected.
2. Returns the address of:
   - The routine to get messages (routine2)
   - The routine to get end-of-type (routine3).

Session recovery: You can select various forms of recovery notification to inform the user that a takeover has taken place. After the alternate system has completed its initialization, this notification takes place.

For more information about terminal switching, see "Terminal Switching" on page 233.

The reconnection process

CICS attempts to acquire terminals defined as AUTOCONNECT(YES) shortly after control is given to CICS.

Autoconnect retries: If you are not using XRF and the attempt fails, there is no retry, and the terminal is not bound. After an XRF takeover, there will be a period in which terminals are physically switched to the new active system, and the network is reconfigured by means of VARY commands. Autoconnect is more likely to fail for temporary reasons. So, after takeover, the reconnect process is run and re-run at increasing intervals, for about half-an-hour, or until all the terminals that are marked to be reconnected are bound.

ZCP workload spreading: The first priority after an XRF takeover is to get the XRF-capable terminals (those terminals with backup sessions) back to work. To avoid interfering with this, the first autoconnect attempt is delayed by up to four minutes, based on the number of standby sessions that needed switching. The user can further extend this delay using the SIT keyword, AUTCONN. Don't confuse the AUTCONN keyword with the AUTOCONNECT option for defining terminals.

Warm starting session-states: Since the state of sessions on the active system is being tracked, it is possible, in effect, to warm start even the session-state of VTAM terminals without backup sessions. The AUTOCONNECT setting for a terminal is regarded as the cold start session-state, and is honored:

- If no tracking is done (that is, at a CICS cold start).
- For warm and emergency restart.
- For XRF takeover for terminals which are defined not to have their session-state tracked.

When the alternate system gets a session-state tracking message for a terminal, it takes precedence over the AUTOCONNECT setting.

There is more information about the reconnection process in "Reconnect processing" on page 237.

XRF tracking streams

Messages are sent by the active system in complete batches called streams. The start-of-stream message identifies a number of types of message in the stream. All the messages in a stream are assumed to be ordered in time. The individual types are ended one at a time. When the last type is ended, the stream is considered to have ended.

All messages carry an identifier. All messages in a particular stream carry the same identifier.

XRF tracking transmits updates on a special-purpose stream, the broadcast stream, which uses a reserved value of identifier (zero), and sends catch-up information using specific streams with non-broadcast IDs.

Each message also carries a key. This gives the position of the message data in the structure.

An interface macro, ZXTR, is defined in copy file DFHZXTR. This encapsulates the code used to send and receive messages in the active and alternate systems. This code calls the XRF message manager to transmit the records.

Figure 63 on page 226 through Figure 66 on page 228 illustrate the description in the text. Keywords are printed in upper case.

Preparing to receive a complete stream

See Figure 63 on page 226. The receiver issues initialize and open for input parameters, using the **nextstream** option to accept messages for the next stream that starts up. Routine1 gets control when this happens, as described below.

The value "B" in the example's ID parameter correlates the **open** to any other requests specifically for this stream, elsewhere in the same module.

The loop containing **run** serves all the streams, and runs until the message manager indicates end-of-data, or an error occurs. When an error does occur, there is no need for error recovery, because streams can be re-sent in their entirety.

```
Sender          .   Message   .        Receiver
                :             :
                :             : (1) Use ZXTR INIT to say that
                :             :     the caller wants to INPUT
                :             :     messages.
                :             :
                :             : (2) Use ZXTR OPEN to open
                :             :     INPUT streams, and supply
                :             :     it with a START routine
                :             :     (routine1 in Figure 64 on page 227)
                :             :     that deals with the start-
                :             :     of-streams message when
                :             :     it arrives.
                :             :
                :             : (3) Use ZXTR RUN to look for
                :             :     incoming messages defined
                :             :     by (1) & (2).
                :             :     Continue until no more
                :             :     messages.
                :             :
```

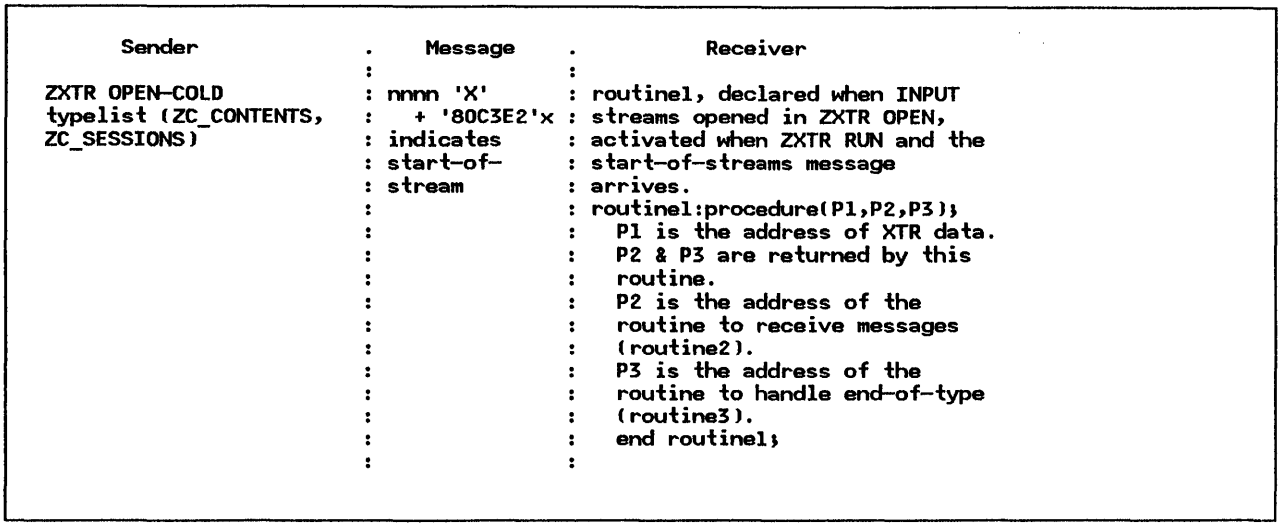
Figure 63. Tracking streams – an example (part 1 of 4)

Start-of-stream

The sender starts a fresh stream using the **open** parameter, as shown in Figure 64 on page 227. The **cold** option declares that this stream is not the continuation of a previous stream, and will be complete in itself. **Nextstream** causes a new stream-ID to be assigned from a global counter.

When the start-of-stream message reaches the receiver, the routine named in the **start** keyword of **open** is given control. This routine:

1. Notes the list of types, so that end-of-stream can be detected.
2. Returns the address of:
   - The routine to get messages (routine2)
   - The routine to get end-of-type (routine3).



```
      Sender            .    Message     .          Receiver
                        :                :
ZXTR OPEN-COLD          : nnnn 'X'       : routine1, declared when INPUT
typelist (ZC_CONTENTS,  :   + '80C3E2'x  : streams opened in ZXTR OPEN,
ZC_SESSIONS)            : indicates      : activated when ZXTR RUN and the
                        : start-of-      : start-of-streams message
                        : stream         : arrives.
                        :                : routine1:procedure(P1,P2,P3);
                        :                :   P1 is the address of XTR data.
                        :                :   P2 & P3 are returned by this
                        :                :   routine.
                        :                :   P2 is the address of the
                        :                :   routine to receive messages
                        :                :   (routine2).
                        :                :   P3 is the address of the
                        :                :   routine to handle end-of-type
                        :                :   (routine3).
                        :                :   end routine1;
                        :                :
```

Figure 64. Tracking streams – an example (part 2 of 4)

Transmitting a message

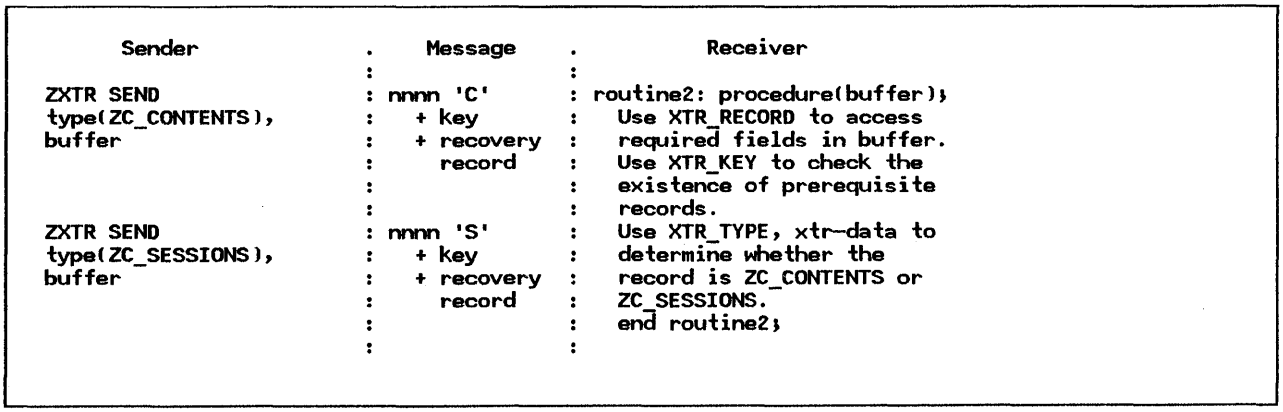
The sender passes the message using the structure **xtr_record** by giving its address in **buffptr**. The send is related to the open for the stream by the ID value. See Figure 65.

The caller is responsible for allocating storage and setting values for the following fields in structure **xtr_record** based in that storage:

- The tracking key is placed in **xtr_key_value**, a character field. Its maximum length is given by a fixed value **xtr_max_keylen**.
- The length of the key goes in the fixed field **xtr_key_length**.
- The message data is passed in **xtr_data_string**, a character field.
- Its length goes in the fixed field **xtr_data_length**.

DFHZXTR gives layouts for **zc_sessions** data, and for control record data.

The receiver gets control in the routine identified by its START routine's second parameter. Its parameter is the **xtr_record** structure.



```
      Sender            .    Message     .          Receiver
                        :                :
ZXTR SEND               : nnnn 'C'       : routine2: procedure(buffer);
type(ZC_CONTENTS),      :   + key        :   Use XTR_RECORD to access
buffer                  :   + recovery   :   required fields in buffer.
                        :     record     :   Use XTR_KEY to check the
                        :                :   existence of prerequisite
                        :                :   records.
ZXTR SEND               : nnnn 'S'       :   Use XTR_TYPE, xtr-data to
type(ZC_SESSIONS),      :   + key        :   determine whether the
buffer                  :   + recovery   :   record is ZC_CONTENTS or
                        :     record     :   ZC_SESSIONS.
                        :                :   end routine2;
                        :                :
```

Figure 65. Tracking streams – an example (part 3 of 4)

Ending a stream

Whenever the sender determines that it will not need to send any more messages of a given type, it issues **end_of_type**. This is illustrated in Figure 66 on page 228. The message is passed to the receiver's end-of-type routine, where the field **xtr_type** of the **xtr_record** structure can be used to remove the type from the list set up by the start routine. When the list is empty, the receiver can issue a close input command to prevent any further calls to the message routine, or the end-of-type routine.

When the sender determines that it has ended all the types in the stream, it can issue a close.

```
      Sender              .     Message     .          Receiver
                          :                 :
ZXTR END-OF-TYPE          : nnnn 'C'        : routine3:procedure(buffer);
(zc_contents), buffer     :   + null key    :   ZXTR CLOSE(input);
                          :   + null data   :   end routine3;
                          :                 :
                          :                 :
                          :                 :
ZXTR END-OF-TYPE          : nnnn 'S'        :
(zc_sessions), buffer     :   + null key    :
                          :   + null data   :
                          :                 :
                          :                 :
                          :                 :
ZXTR CLOSE                :                 :
                          :                 :
```

Figure 66. Tracking streams – an example (part 4 of 4)

How streams are used

Catch-up streams: A catch up is a complete stream, started cold. In the example in Figure 67 on page 229, all the messages carry the same ID of "1".

DFHZXCU first reads the ZCP keyrange of the CICS catalog, using code mostly copied from CICS restart. The XRF tracking key is constructed, and the catalog record sent as the data.

Note: The data also contains the catalog key, the value of which usually differs from the tracking key for a variety of reasons.

The catalog key is obtained from ZC INSTALL, using the DHZCP FUNCTION(MAKEKEY) macro. This is converted to an XRF tracking key using the ZXTR(MAKEKEY) macro.

When the last catalog record has been processed, DFHZXCU sends end-of-type for TCT-contents. This indicates that the alternate system now has a complete image of the contents of the TCT.

Then DFHZXCU scans the VTAM-supported TCTTEs, using code adapted from warm shutdown (DFHWKPZC). Each TCTTE is passed to DFHZXST, which checks the bound/unbound indicators, the XRF-capable flag, copies the correlation ID, and so on, and notifies the alternate system, as for normal tracking.

In the alternate system, DFHZXST unpacks the message, and applies its contents to the TCTTE, starting standby-BIND processing, if needed, as for normal tracking.

When the last VTAM-supported TCTTE has been processed, DFHZXCU sends end-of-type for ZCP-sessions. This indicates that the alternate system now has a complete image of the state of the active's sessions.

Note: DFHTCRP throws away catch-up messages if a tracking message with the same key-value has already been processed, or if a catch up has already been done for this alternate system.

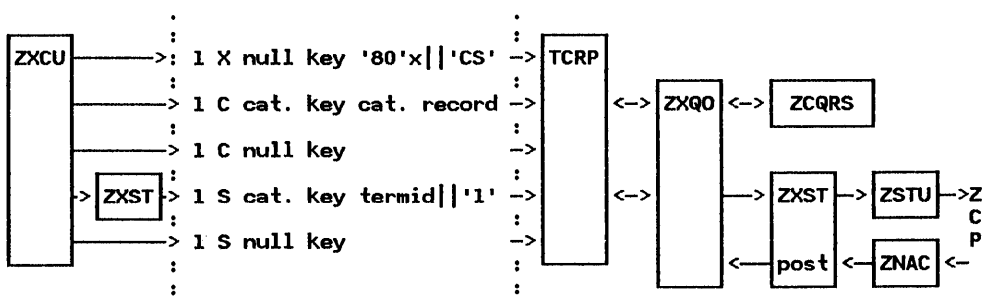


Figure 67. Catch up

The broadcast stream: All updates are done as a single stream, using the special-case ID of zero – the broadcast stream. This is started in DFHTCRP and ended in DFHWKP, as outlined in Figure 68.

As soon as an active system gets control of the catalog, DFHTCRP notifies any alternate systems by sending a broadcast start-of-stream. This contains a list of the types that will be in the stream. The stream is considered closed when an end-of-type message has been sent for each one of these. This happens if the active system is shut down normally, using DFHWKP.

Any changes to tracked resources are notified to the alternate system using broadcast messages with the appropriate type code, as explained below.

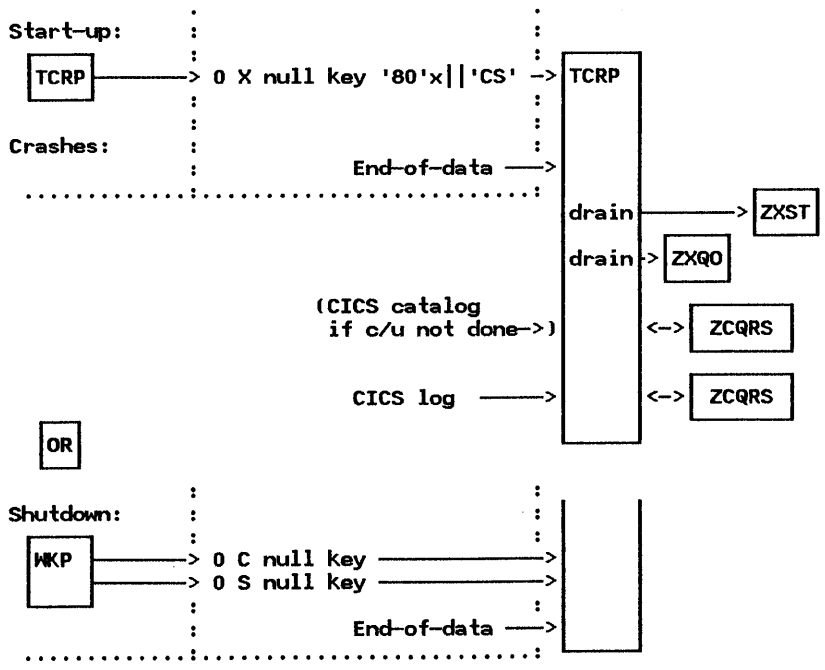


Figure 68. Tracking (start and end)

Tracking streams: The messages to notify updates are raised at a variety of points in CICS, as shown in Figure 69 on page 230.

The TCT-contents records are processed in the alternate system by the ZC RESTORE code that reinstates TCT entries during CICS restart.

The ZCP session-state records are processed by DFHZXST, as described below.

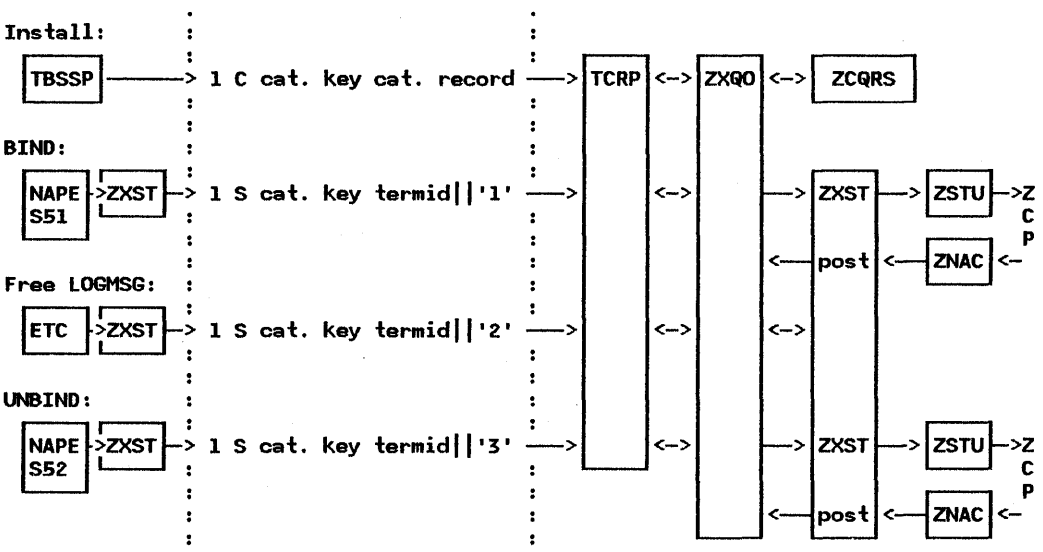


Figure 69. Tracking

The queue organizer – DFHZXQO

The interface to the queue organizer is:

- INITF – initialize for start-of-day.
- ADDACT – add the given action to the relevant node of the internally-maintained tree. The action is initiated if no other actions are held up.
- POST – tell the queue organizer that a previously initiated action has now completed.
- DRAIN – remove from the tree actions held up by a current action that has not posted the queue organizer.

The only user of ADDACT is DFHTCRP, which sets a parameter block depending on the type of tracking message received. If it received a "C" type message, for example, zc_contents for install or delete, it sets a parameter block with the address of the routine DFHTCRPC. For an "S" message, for example, zc_session for bind or unbind, the routine address is DFHTCRPS. See "XRF tracking streams" on page 225 for more information about these messages.

When ADDACT is invoked, the queue organizer checks for prerequisites, and, if none exist, calls the routine specified by DFHTCRPC or DFHTCRPS. This resource manager returns **done**, **scheduled**, or **error**, which eventually results in the DFH1022 error message. The scheduled return code tells DFHZXQO to expect a post to be issued in the future. Other actions related to this entity (as defined by the key) are held.

When the post is issued by DFHZNAC, DFHZXQO frees the work space associated with the posted action, and looks for more actions to start.

Standby BIND processing

Establishing a session

As soon as a new session is established in the active system, the alternate system is notified (unless the terminal is defined with RECOVERYOPTION(NONE)), by code in DFHZNCA.NAPES51. The message to the alternate system includes any logon data, and a BIND-image if the TCTTE is defined with the LOGMODE keyword.

In the alternate system, DFHZXST sets the flag TCTEXRE to record the known state of the active session, and, if the active session was XRF-capable, starts the process of acquiring a backup session. See Figure 70.

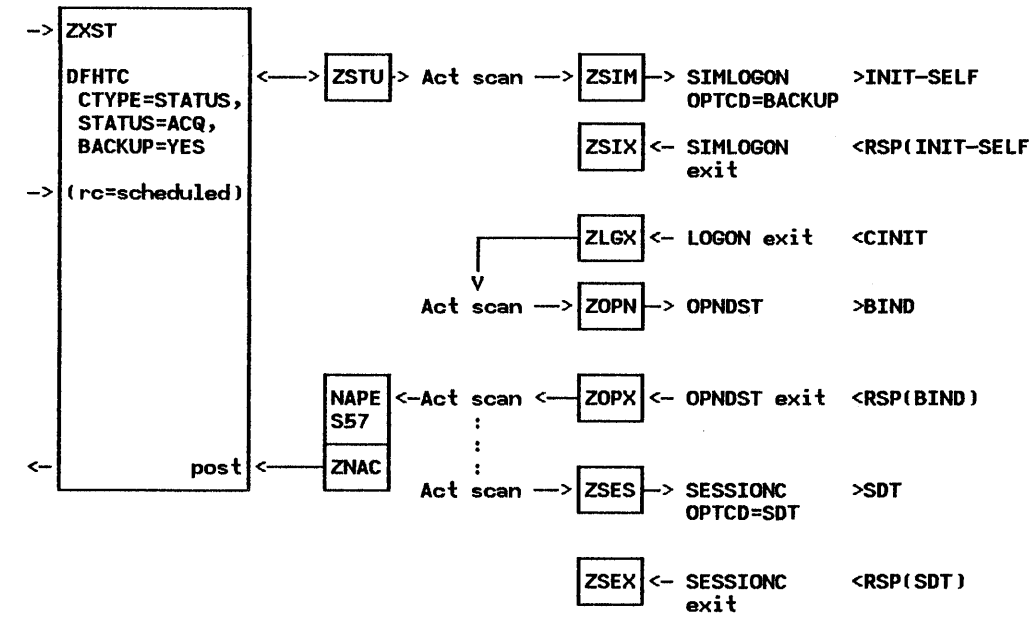


Figure 70. Establishing a standby session

If the session does not require a backup session, or the alternate system fails to get a backup session, then TCTEXRE is detected by the autoconnect transaction, in program DFHZXRE0, and a session is acquired as for autoconnect.

The keyword on DFHTC CTYPE=STATUS sets a TCA request flag for DFHZSTU, which is passed to DFHZSIM using a TCTTE flag, TCTEXSB. If this is set, DFHZSIM sets the flag in the RPL for OPTCD=BACKUP, which causes VTAM to drive DFHZLGX in the usual way. OPTCD=BACKUP takes effect on the subsequent OPNDST.

For a standby-bound TCTTE, the flag TCTEXCS is set in addition to the usual TCTEOPD flag. This is set by DFHZSIM when requesting the session.

DFHZLGX disallows the NIBSRCH (which uses NETNAME to find a TCTTE) during XRF tracking. Then DFHZSIM passes the TCTTE address using the RPL user field, in the normal way. This has the effect of preventing the end user from logging on directly to the alternate system.

The CINIT received in the logon exit, DFHZLGX, indicates the XRF-capability of the session requested – that is, its capability of being XRF switched. DFHZLGX copies this indication into a TCTTE flag TCTEXCA. This flag is cleared whenever it determines that CICS is unable to support the session as XRF-capable for any reason.

DFHZLGX then queues the TCTTE for DFHZOPN as normal.

When DFHZOPN detects TCTEXCA, it issues an XRF-capable BIND, which differs from the normal BIND in that it carries the XRF control vector. This includes a security key – the correlation ID. The value of this key is devised by the active system and passed to the NCP boundary function on the active BIND. The alternate system must supply the same value of key on its standby BINDs, before the boundary function in the NCP will accept them. To implement this scheme, if DFHZOPN detects TCTEXCA, but not TCTEXCS, it attaches the value of the correlation ID to the TCTTE. The value is generated using the STCK instruction and saved in the LUC extension for LUC TCTTEs (field TCTESIF), or in an extension addressed from the LUC extension anchor for non-LUC TCTTEs.

The correlation ID is picked up by DFHZXST, and included in the message sent to cause the alternate system to issue standby BIND. DFHZXST in the alternate system attaches a copy of the correlation ID to the TCTTE. Whichever way the correlation ID is set up, TCTEXCA causes DFHZOPN to append it to the BIND in the XRF control vector.

OPNDST completion is done in the DFHZNCA exit NAPES57. This is an abbreviated version of NAPES51. It is worth noting that the normal NAPES51 processing should be logically equivalent to standby-BIND completion followed by SWITCH completion.

As soon as DFHZXST has issued the DFHTC CTYPE request, it returns control to DFHZXQO with a return code that means **scheduled**. This causes DFHZXQO to queue further tracking for this terminal, until DFHZXST makes a later **post** call to it. This call is correlated to the previous **schedule** by a ZXQO **token**, a four-byte value passed to DFHZXST with the **schedule** request, and saved in the TCTTE at TCTEXTOK. The posting is triggered by DFHZNAC, which detects an idle TCTTE with a non-zero value in TCTEXTOK, and passes it to DFHZXST.

DFHZSIM and DFHZLGX also maintain a count of outstanding standby SIMLOGONs in field TCTVXPLC, and there is an ECB, TCTVXPLE, which is **posted** when TCTVXPLC is moved to zero. This ECB and count are examined by DFHZXST during drain processing. The takeover may be held up briefly to allow these to clear themselves. See messages DFH6480 and DFH6475.

NAPES57 and NAPES52 also maintain a count of outstanding standby sessions in TCTVXSBC. This is used to calculate the period by which the reconnection transaction CXRE (program DFHZXRE0) is deferred, to a maximum of four minutes. This allows the XRF-capable terminal recovery to take priority.

Ending a session

Whenever a session in the active system is unbound, CLSDST completion (NAPES52 in DFHZNCA) notifies the alternate system. The NCP boundary switch also notifies the alternate system that the active session has gone. (See Figure 71 on page 233.) The notification that reaches the alternate system first causes CLSDST processing to start. In the time this takes to finish, the second notification may arrive.

The schedule/post mechanism outlined above is used to delay tracking activity for the terminal until the session is unbound, in particular, the deletion of the TCTTE caused by a transient (autoinstalled, for instance) terminal logging off.

Two possible flows can occur when ending a session. This is because a race condition can occur between a hierarchical-reset condition from VTAM, and a session-tracking message from the active system which indicates unbind required. Figure 71 on page 233 shows the latter occurence. DFHTCRP receives the unbind request, which is passed to DFHTCRPS (under control of DFHZXQO). This calls DFHZXST, which checks the status of the TCTTE. If it finds there is still a session, it initiates a CLSDST, and returns **scheduled** to DFHZXQO. It also saves the DFHZXQO token in the TCTTE for later use.

The hierarchical reset then comes in, triggering DFHZNAC with error code X'3A'. At label NACPEND, DFHZNAC calls DFHZXST post, which checks whether the TCTTE is stable, or whether there is still some work to be done; that is, it looks at the action flags in TCTEACR. It does this only if there is a DFHZXQO token in the

TCTTE. This first call to DFHZNAC contains action flags, indicating CLSDST.

Next, the CLSDST completes, and DFHZNAC is triggered again with error code X'49'. NACPEND is reached, and an attempt is made to post DFHZXST, but the post cannot be done because the action flags indicate FREEMAIN.

The active scan checks the token field in the TCTTE, and the action flags. If no action flags are set, DFHZNAC is triggered with error code X'00'.

This time NACPEND is reached without doing anything, and, because no action flags are set, it ends DFHZXST, and calls the queue organizer, using post.

The second case is when the hierarchical reset occurs first. This means that when the tracking message arrives, DFHZXST discovers, on examining the TCTTE, that there is no backup session, so it returns **done** to the queue organizer. DFHZNAC does not perform a post, because there is no token in the TCTTE.

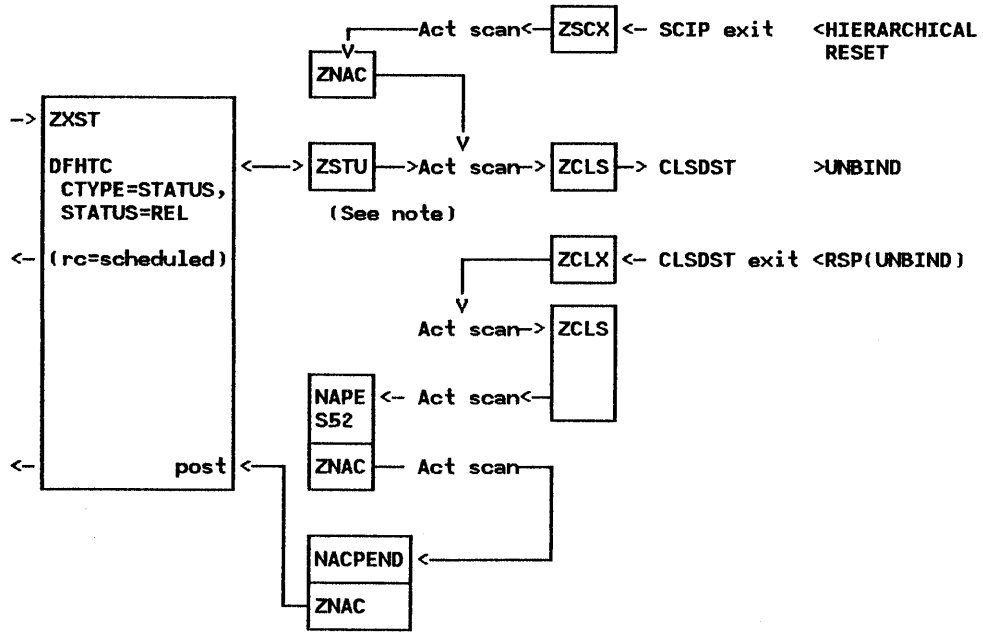


Figure 71. Ending a standby session

Terminal Switching

Refer to Figure 72 on page 234.

DFHXTCI is invoked when the alternate system is requested to begin the takeover. This program scans the TCT to locate the TCTTE entries that have been bound with backup sessions.

The entries with backup sessions are passed to DFHZSES. DFHZSES issues the SESSIONC CONTROL=SWITCH command. The data that is returned in response to the SWITCH command details the last data flow that passed the switch point.

The SWITCH response data is analyzed by DFHZXRC, which determines the state of the session. DFHZXRC initiates the action that is necessary to clean up the session, and then passes the terminal to DFHZNAC for the conversation restart processing.

DFHZNAC checks and initiates the user's requested recovery notification procedure.

The terminal is then returned to service.

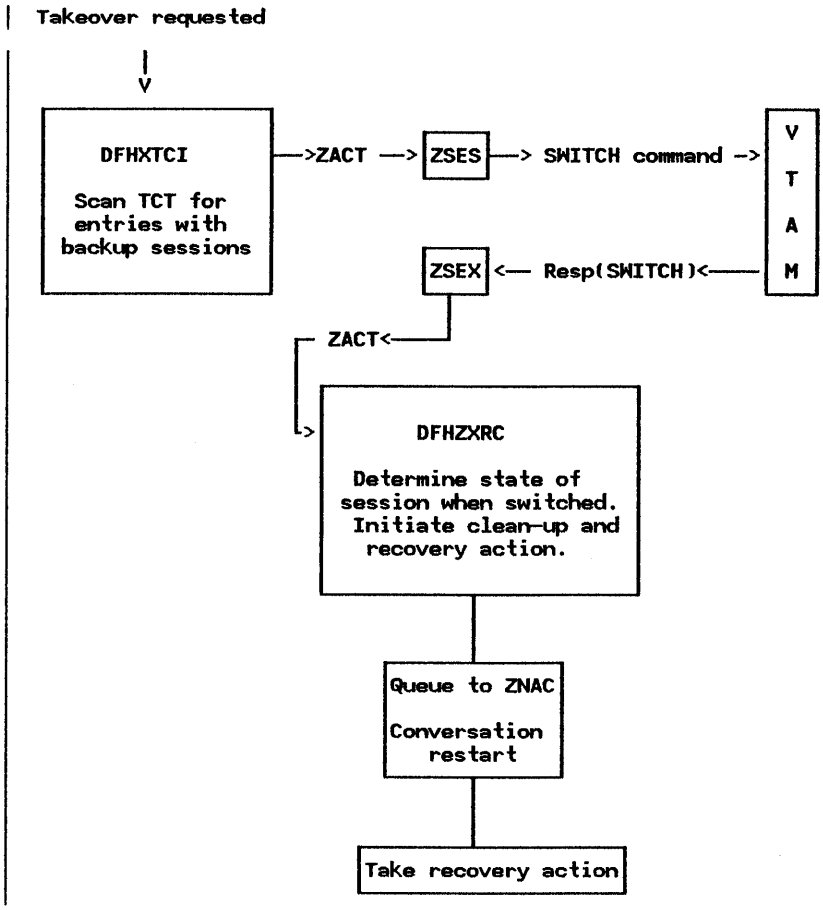


Figure 72. Terminal switching overview

Session state analysis

The data received in response to the SESSIONC CONTROL=SWITCH command is described in the architected session state data vector X'29'. This vector details the last data that passed the switch point before the SWITCH command was issued. The data is analyzed by module DFHZXRC in order to determine the action to be taken to clean up and recover the session, so that the end user can continue operating. The vector is described in the DSECT DFHTCV29.

Refer to Figure 73 on page 235.

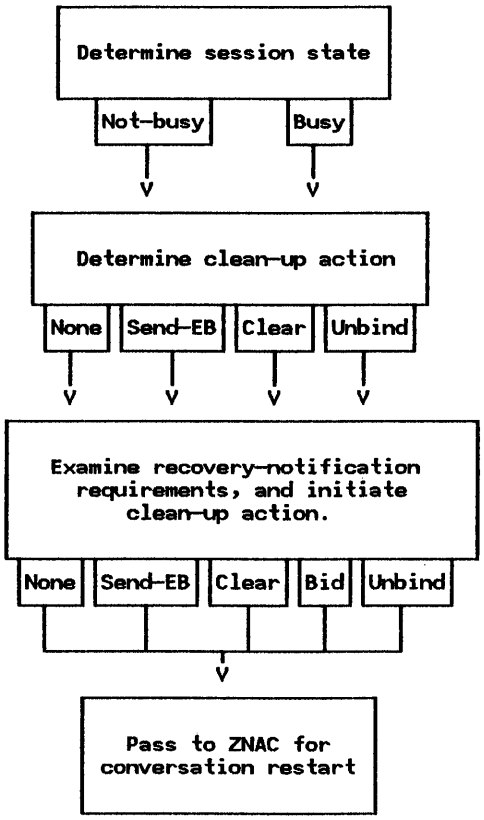


Figure 73. Session state analysis

Clean-up action

If the terminal was not in use at the time of the switch, the sessions are not busy, so no action is required. If the terminal was processing a transaction when it was switched, the session is busy and DFHZXRC has to take action to clean it up.

The session is judged not to be busy if the response data received to the SWITCH command shows that the last flow(s) between the active system and the terminal is one of:

- A request sent RQE with the end-bracket indicator.
- A definite response to data that was sent with an end-bracket indicator.
- A definite response to a start data traffic command.

The actions DFHZXRC can take to clean up a busy session are:

- Send an end-bracket indicator.
- Send a clear command.
- Send an unbind command.

The choice of actions depends on:

- The exact state of the session when it was switched, for example, is the terminal in receive mode or send mode?
- The session characteristics, for example, is **clear** supported?

- The terminal definition parameters. for example, what recovery option was requested?

Recovery action

Before DFHZXRC initiates the necessary clean-up action, it examines the user's requested recovery notification. This recovery notification is not initiated until after the DFHZNAC conversation restart processing, but DFHZXRC can modify its immediate clean-up action in order to prepare for the subsequent recovery notification action.

The recovery notification options available are:

- None.
- Send a recovery message.
- Initiate a recovery transaction.

Examples of resultant modification to the basic clean-up action are shown in Figure 74.

```
IF clean-up action = send end bracket
AND
recovery notification = send recovery message
THEN
Take no immediate action, because the recovery message will be sent
with the necessary end bracket.

IF clean-up action = none
AND
recovery notification = send recovery message
THEN
issue bid immediately so that the recovery message can be sent with
begin bracket and end bracket.

IF clean-up action = send end bracket
AND
recovery notification = initiate recovery transaction
THEN
send end bracket and issue the bid command ahead of the subsequent
transaction initiation request.
```

Figure 74. Clean-up action modification

DFHZNAC – conversation restart

After the alternate system has been fully initialized, each terminal that was switched is passed to DFHZNAC for conversation restart processing. This is similar to the session-opened processing for a normal session start.

The action of DFHZNAC is:

1. Execute normal **session-just-opened** logic – NAPES51
2. Invoke a user's NEP. Parameters passed include:
   - Recovery notification
   - Recovery message details
   - Recovery transaction details.

   The NEP is able to change these parameters.
3. Perform the recovery notification.

Recovery notification

The purpose of recovery notification is to notify the end user that service has returned. Depending upon the user's working environment, notification may not always be required, in which case the takeover may be completely undetected.

Depending upon the requested notification, the recovery action of DFHZNAC is:

- If none – pass to DFHZDET to clean up the TCTTE.
- If recovery message – check the bracket state, and issue DFHBMS send map.
- If a recovery transaction – issue DFHIC initiate for the transaction.

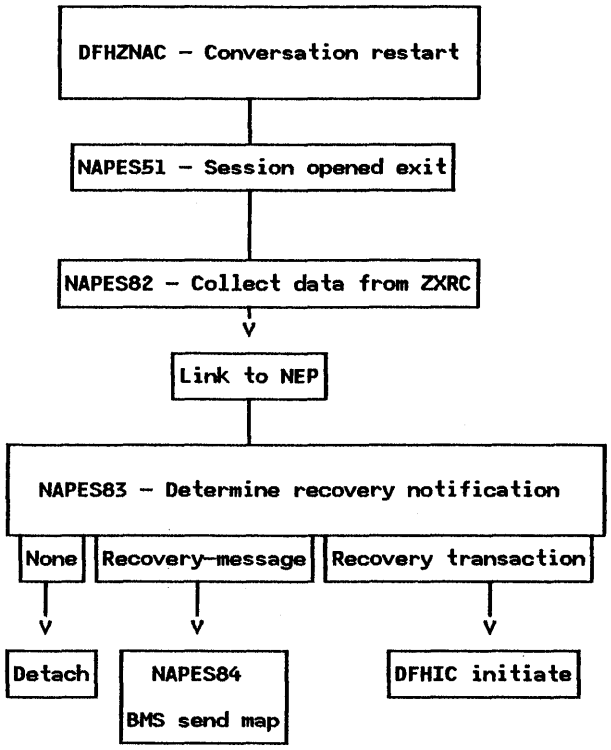


Figure 75. DFHZNAC – conversation restart

Reconnect processing

Reconnect processing is under the control of DFHZXRE0. This is run under the transaction CXRE, as a non-terminal task. It is first scheduled by DFHSIJ1 after control is given to CICS, and, after takeover, reschedules itself at progressively greater intervals for about half an hour.

DFHSIJ1 calculates a delay of up to four minutes from the number of standby BINDs at the time of the takeover, and adds a further delay specified by the user in the AUTCONN parameter of the SIT. The result is sent to the operator in message DFH6481 if the result is not zero.

The terminal control table is scanned using the CTYPE=LOCATE macro, and all terminals and sessions which are flagged to be (re)connected (TCTEXRE) are passed to DFHTC CTYPE=STATUS,STATUS=(INSRV,ACQUIRE).

It replaces the similar function in DFHZSLS, but only runs itself once if the start-up is not an XRF takeover. In this case, flag TCTEXRE is initialized from the bit in the AUTOCONNECT parameter, TCTIQSL. If there is no tracking, this setting is acted upon by DFHZXRE0. However, if a session-state tracking message arrives for this TCTTE, the setting of TCTEXRE is overwritten.

The pass-number is maintained in TCTVXREN. This indicates the number of times the new active system tries to autoconnect VTAM sessions after takeover, displayed by message number DFH6490I)

MODIFY USERVAR

The takeover is intended to be transparent to the end user (subject to the limitations of the terminal in use), and it is clearly desirable that users can log on with an application ID value which stays the same across a takeover.

Overview

Release 3.1 of VTAM introduces the concept of a USERVAR. This provides a mechanism whereby a single application ID value can be used to log on to different applications at different times, under control of the system operator. For example, **LOGON APPLID(CICS)** can cause logon to **CICSA** or **CICSB** at different times.

The operator may set different values of this mapping by a MODIFY command which would have the form:

```
F procname,USERVAR=CICS,VALUE=CICSB
```

in the above example.

CICS also issues these commands using the operating system MGCR macro, to ensure that logons are directed to the intended system, and to re-route logons to the new active system when an XRF takeover occurs.

Generic and specific APPLIDs

In an XRF complex there may be two CICS systems which are the target of logons using a single APPLID value. This single value represents the complex as a whole, and is called the generic APPLID. Each possible active system has its own APPLID value, called the specific APPLID. At any one time, only one of these is the setting of the USERVAR for the generic ID.

In XRF, the DFHSIT macro and the SIT overrides accept values for both the generic and specific APPLIDs.

When a system is brought up with XRF = NO, it modifies the USERVAR for its generic USERID to have the value of its specific USERID. This has the effect of claiming all logons to the generic ID. In this case, the generic and specific IDs may well have the same value. For safety, the MODIFY is re-issued just before the ACB is opened as a result of a CEMT SET VTAM OPEN command.

When a system is an active system brought up with XRF = YES, it modifies the USERVAR for its generic USERID to have the value of its specific USERID, with the same effect of claiming all logons to the generic ID. In this case, however, the generic and specific IDs should not have the same value. If the alternate system starts to take over, it modifies the USERVAR for its generic USERID to have the value of its specific USERID. With the same effect of claiming all logons to the generic ID, this prevents further logons to the failing active system. New logons are queued to the new active system, which accepts them as soon as takeover is complete.

For safety, the new active system reissues the MODIFY, when control is given to the new CICS, and also just before the ACB is opened as a result of a CEMT SET VTAM OPEN command.

Issuing MODIFY USERVAR

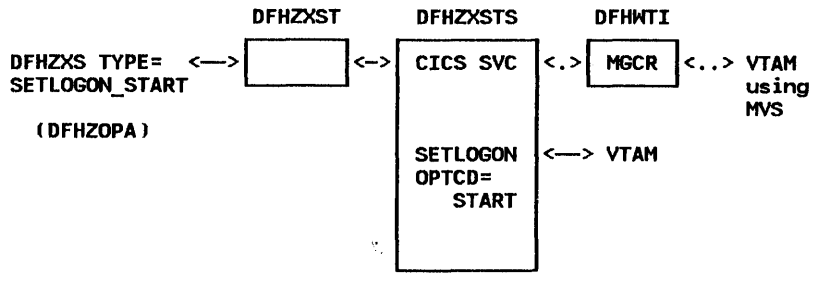


Figure 76. Issuing F USERVAR for SETLOGON START

As shown in Figure 76, the MODIFY USERVAR command is issued as part of the SETLOGON START process when control is given to CICS, and also when the VTAM ACB is opened during CEMT SET VTAM OPEN.

DFHZXSTS issues the SETLOGON and also invokes DFHWTI to pass the "F" command to the operating system.

The original SETLOGON in DFHZSLS remains, but is only issued when an alternate system is starting. DFHZSLS runs as the first process in the new TCP task, which is now entered earlier, in DFHSII1.

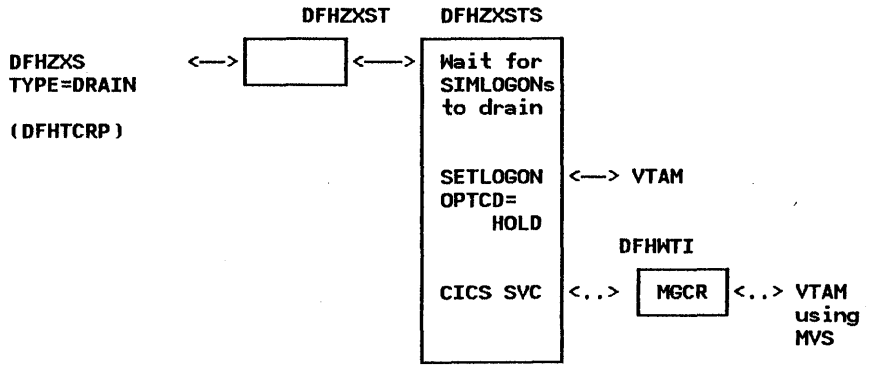


Figure 77. Issuing F USERVAR for DRAIN processing

Figure 77 shows the flow of control for DRAIN processing. This occurs during takeover, to allow all the standby SIMLOGONs that have been issued to appear back in the logon exit, before the logon exit is disabled. An apparent holdup raises the messages DFH6480 and DFH6475.

To route new logons away from the failing active system, MODIFY USERVAR is issued straight after the SETLOGON HOLD. Further logons are suspended until the above SETLOGON START is issued, when control is given to CICS.

DFHWTI

DFHWTI is invoked using the CICS SVC for integrity reasons. It recovers the generic and specific IDs that it builds into the "F" command from the CICS authorized facility control block (DFHAFCS). This is attached to the CICS TCB in protected storage while CICS is still running authorized.

DFHWTI obtains the VTAM **procname** by scanning the MVS CSCB chain for one that has the VTAM address space ID (in ISTAVT). DFHWTI obtains the required MVS ENQ on the CSCB chain during this scan. Although the user could, in theory, substitute a different job with the same **procname** in the time between the DEQ and the MGCR macro, this is unlikely.

If the MODIFY fails?

VTAM cannot tell CICS if the MODIFY fails. This limits the value of trying to handle failure of the MGCR, or even the CICS SVC, within DFHZXSTS or its caller. The most recent errors are reported, though there is little that CICS can reasonably do. The user can re-issue the MODIFY from the operator console.

The XRF overseer

The XRF overseer provides:

- CICS services (running in a non-CICS environment) that build an access structure for, and extract data from, the surveillance file for active and alternate pairs of systems, and return this data. See the *CICS/MVS Customization Guide* for a description of these services.
- A CICS supervisor state function to submit MVS commands, issue JES JOBID status inquiries, and issue JES job cancels.

A sample program is provided that enables the operator to monitor the state of the active and alternate pairs. The user can customize the sample program to add extra function. The *CICS/MVS Customization Guide* gives details about the sample program, and the *CICS/MVS CICS-Supplied Transactions* describes the operator commands that accompany the program.

CICS authorized supervisor state requirements

CICS authorized address space initialization

The //EXEC statement selects the authorized DFHWOS module in SYS1.LINKLIB or STEPLIB (or another installation-authorized library). DFHWOS then transfers control to DFHWOSA (in DFHLIB) to:

- Check the // EXEC statement parameters for the IOPN=OVERSEER PROGRAM NAME keyword. Other keywords are described in the DFHWOSA prologue.
- Ensure the address space is authorized.
- Build the supervisor state entries and latent parameters required to issue MVS command and JES functions.

DFHWOSA then loses authorization, links to the named overseer program (with no DCB specified), and passes the address of the address-space latent parameter in register 1 to the overseer program. The overseer program saves it, to pass it on as an argument for DFHWOSM. The DFHWOSM build function completes initialization, and returns the address of the overseer services parameter area (OSP) in register 1. This is used for the other DFHWOSM service calls.

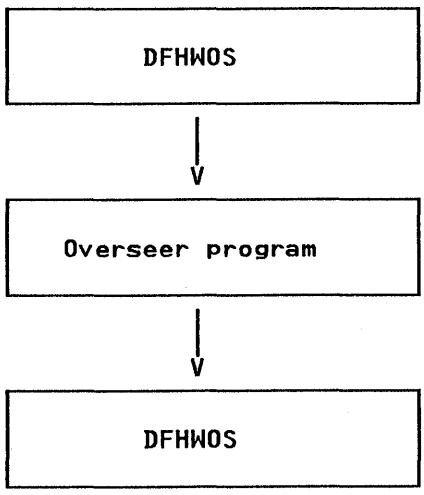


Figure 78. Relationship between DFHWOS and the overseer program

The DFHWOSM macros that define the CICS systems to the overseer program are described in the *CICS/MVS Customization Guide*. In this manual, the actions of the overseer service programs invoked by those macros are described.

DFHWOSM FUNC=OSCMD|JJS|JJC

For authorized requests, DFHWOSM FUNC=OSCMD|JJS invokes a CICS service to:

- Submit an MVS command.
- Submit a JES status inquire command to see if a job is running.

  The JES subsystem options block (SSOB), and its array, are copied into user storage for the caller to analyze return codes.

The following authorized functions are provided for an unauthorized overseer program:

DFHWOSM FUNC=OSCMD
Submit an MVS command (using the MVS SVC interface inside the CICS SVC).

DFHWOSM FUNC=JJS
Submit a JES status inquire command to see if that job is running.

DFHWOSM FUNC=JJC
Submit a JES job cancel command.

These functions cause DFHWOSB to be entered using a program control instruction:

- DFHWOSB is loaded at initialization time from an authorized library into non-fetch-protected key 0 storage.
- The program-control latent parameter established during initialization is used to access internal control structures.
- The parameters passed by the caller are copied from user-key storage to key 0 storage for validation.

The overseer program passes the address of the latent parameter passed to it on entry as the token for all authorized state services. The authorized routines always issue a program transfer instruction to return control to the overseer in problem state.

DFHWOSM FUNC=BUILD

An overall surveillance structure is built by this service. The address-space latent parameter is passed as the token operand. A token (OSP) is returned in register 1 to identify the construct; the token is passed on subsequent unauthorized requests. Register 13 points to a set of two chained save areas. The DFHWOSM FUNC=BUILD macro invokes the build function to:

- Build an asynchronous work element (AWE) pool for subsequent **open** requests, and for its own use.
- Set the asynchronous work element queue header(AWEQH) between itself and DFHWOSB to **no-work**.
- Attach an MVS task, using PWT in DFHWOSB as entry point, and passing to it the address of AWEQH. An EXTR routine is used for the subtask.
- Save the address of the attached TCB in the OSP block.
- Set up the extract overall structure and return a build token identifying it to the caller.
- Build a set of generic APPLIDs to null status. A pointer to GENHOLD is defined to track prior read values (none so far).
- Build a set of ACB pairs for the generic APPLIDs, specifying control-interval sequential processing, direct addressing, input mode, and user buffering.
- Build a set of 32K-maximum size VSAM buffers with an error-message area, an internal work area, and a VSAM RPL for each buffer. This set is addressed from the main structure.
- Capture the MVSNAME and MVSIPLTIME, and save it in the main structure.
- Access the authorized routines to ensure they are working properly.

Note: The DFHWOSM FUNC=DSECTS describe the user-key blocks for the overseer program if it needs to access them.

The subtask with entry point PWT in DFHWOSB initializes itself, and then attempts to idle the AWEQH shared between DFHWOSB and itself, find no work to do, set the AWEQH to **idle** (negative address of its ECB), and MVS-wait on its ECB (at this point, an enquiry of work on the AWEQH MVS-posts the subtask).

The subtask performs the following asynchronous requests:

- Terminates itself by returning to the MVS dispatcher.
- Issues a queued JES request (STATUS|CANCEL).
- Issues an MVS timer whose exit breaks long-term waits.
- Issues a VSAM read request. If the read completes, the timer is cancelled, the requestor return code (AWERC) is set, and the requestor is posted.
- If the timer exit is driven, purges the I/O for the VSAM read (SVC 16 or 33), sets the requestor return code (AWERC), and posts the requestor.

The consistency of the CIs read is checked using the subtask.

Output:

1. Return code in register 15:
   - 0 – structure built successfully
   - 4 – token not the latent parameter for the address space
   - 8 – insufficient storage.

   For return code = 0, register 1 contains the **open-token** for use in services such as OPEN, and READ.

DFHWOSM FUNC=TERM

Overseer services are terminated by this service. The token is the build token, and register 1 has the address of the parameter list with the address of the latent parameter passed to the overseer program on entry for this request.

The DFHWOSM FUNC=TERM macro invokes the termination function to:

- Request that the asynchronous task terminates. This service waits until the subtask with entry point PWT has stopped processing requests, and is going to return to MVS.
- Close any open surveillance file(s) and release their storage.
- Wait until the EXTR has been scheduled under its TCB for subtask with entry point PWT, before it detaches the subtask TCB. The routine then returns to the caller.

  Note: If the detach is not done before the caller's subtask goes away, a detach abend will occur. (This can happen on abends.) The caller can have an ESTAE issue the termination macro to stop detach abends.

Input parameter list pointed to by register 1:

1. Address of the token created by build.

DFHWOSM FUNC=OPEN

The build token is passed as the token operand. The user's call list includes: the address of the GEN-APP, the address of the ddcard for the STATE data set, and the address of the ddcard for the MESSAGE data set. The routine returns a code in register 15. (A return code of 0 through 5 is a viable state to continue. A return code of 8 or more is not satisfactory.) Register 13 points to a set of two chained save areas. The open service serially uses the structure made by the build function, and returns to the caller after the open function has completed.

The OPEN service:

- Ensures the GEN-APP, or the ddnames, do not already exist in an opened GEN-APP set.
- Allocates a generic APPLID, and a pair of ACBs (state-message file) from the build construct, fills them in from the user's parameter list, and:
  - Opens the pair of ACBs for the surveillance generic APPLID.
  - Issues a SHOWCB for HALCRBA, CISIZ, LRECL, and BLKSIZE, and ensures that they are satisfactory. Then saves the block size of the state, and message data sets, in the generic APPLID.

    One string is sufficient to read CIs serially. Up to three strings for a data set could be active for three different CI read requests. This would require that three internal ECBs be posted by VSAM, before all requests have completed, and the extract service finishes its work and returns to the caller.

    An extra area for the GEN-APP is allocated, so the previous read buffer content can be compared with the current buffer content to determine the state.
- Opens the ACBs under the caller's TCB. If the data set is empty, the ACB OPEN request accesses the VSAM catalog, to ensure the ESDS data-set extent has been formatted, and causes the VSAM OPEN to fail.
- Get an AWE from its pool to communicate with the read subtask.
- Requests the asynchronous task to read the state and message data sets by enqueueing an AWE on its AWEQH.
  - A one minute timer is set in case the read does not complete.
  - The state and message data set CIs are read in.
  - The timer is canceled.
  - Certain fields in the CIs must be in a consistent state – that is, they must not have been changed by another program while the CIs were being read. A global count field of sign ons done for the generic APPLID is saved from the first CI read. The remaining CIs are read. Then the CI with the global count is reread to check that no other sign ons have incremented the count while the CI reads were in process.

    If a change has occurred, the read is retried to three times. If the CIs are still inconsistent, set the AWERC to indicate that.
  - If the timer exit is driven, purge the VSAM read I/O, and set the return code to indicate a time-out.

    Note: If this is an extract surveillance signal only request, the I/O is purged, and the other VSAM data set is read for the surveillance signal.
  - The AWEECB is posted to wake up the requestor.
  - The MVS subtask successfully idles its AWEQHDR, and waits for further work.
- Checks the generic APPLID files, now that it is back under the caller's TCB.
  - Both the state and message files must have been successfully read.
  - The GEN-APP given must be present.
  - The MVS IPL time must be less than the time a job in the CEC, in which the overseer is running, signed on to the generic APPLID.
- Does not extract any data for the caller.
- Sets the AWERC value into register 15.
- Returns to the caller.

The open service caters for surveillance files in the following states:

- Active and alternate systems signed on.
- Active or alternate system signed on.
- Not in use now.
- The same MVS (SMF Name), but time before MVS IPL.
- Never used before (possibly unformatted).

The token is the build token, and the input parameter list pointed to by register 1 is:

1. Address of the GEN-APP
2. Address of the ddname for the message data set
3. Address of the ddname for the state data set.

Output:

1. Return code in register 15:
   - 0 – surveillance file opened, active and alternate systems signed on.
   - 1 – surveillance file opened, active system is signed on.
   - 2 – surveillance file opened, alternate system is signed on.
   - 3 – surveillance file opened, no XRF activity (not signed on).
   - 4 – the same SMF MVS name, active system before IPL time.
   - 5 – the same SMF MVS name, alternate system before IPL time of the system in which the overseer is running.
   - 6 – CI data not in a consistent state after three attempts.
   - 7 – the GEN-APP given is not in the data set specified by the ddname.
   - 8 – the GEN-APP or ddname is already in the GEN-APP set, no duplicates allowed.
   - C – the MVS open ACB failed.
   - 10 – the SHOWCB for an ACB failed.

DFHWOSM FUNC=CLOSE

The caller sets the token to the build token, and puts in register 1 the address of the parameter list containing the address of the GEN-APP. The service MVS-closes the ACBs for the generic APPLID, cleans up the generic APPLID, and sets a return code in register 15. Register 13 points to a set of two chained save areas.

The DFHWOSM FUNC=CLOSE macro invokes the close routine to:

- Ensure the GEN-APP matches the structure.
- Close the ACBs.
- Release the generic APPLID entry.

Output:

1. The return code set in register 15 is:
   - 0 – close successful.
   - 4 – token unrecognizable.
   - 8 – GEN-APP not in generic APPLID set.

DFHWOSM FUNC=READ

CICS feeds back information extracted from the surveillance files for a given generic APPLID. The token is the build token, and register 1 points to a parameter list containing the address of the GEN-APP, and the address of an area containing DBLLIST with entries selecting which items are to be returned. The DBLLIST is a set of double-word entries describing each item that can be accessed by the extract service. The caller sets the type field of each entry, the service sets the length and address of each item as it processes the DBLLIST. Register 13 points to a set of two chained save areas. Register 15 is set with a return code. The DFHWOSM FUNC=READ macro invokes the extract service. The service caters for files being in the following states:

- Never used before (possibly unformatted).
- Neither the active system nor alternate system is signed on and:
  - The active and alternate systems signed off normally.
  - The active and alternate systems signed off abnormally.
  - One system signed off normally and the other abnormally.
- The active system alone is signed on and:
  - The alternate system never signed on.
  - The alternate system signed off normally.
  - The alternate system signed off abnormally.
- The alternate system alone is signed on, and:
  - The active system never signed on.
  - The active system signed off normally.
  - The active system signed off abnormally.
- The active and the alternate systems are signed on, and:
  - They are running normally.
  - The active system is in trouble.
  - The alternate system intends to become active, and is not incipient yet.
  - The alternate system is in trouble.
  - The alternate system has become the incipient active system.
  - The alternate system has become the current active system.

The token is the build token and the input parameter list pointed to by register 1 is:

1. The address of the GEN-APP.
2. The address of the user's DBLLIST entry set describing the item types that are to be retrieved.

Output:

1. Return code:
   - 0 – surveillance files satisfactory, and both active and alternate systems signed on.
   - 1 – surveillance files satisfactory, and active system signed on.
   - 2 – surveillance files satisfactory, and alternate system signed on.
   - 3 – surveillance files satisfactory, but neither active nor alternate system signed on.
   - 4 – same MVS SMF name, but active system before MVS IPL time.
   - 5 – same MVS SMF name, but alternate system before MVS IPL time.
   - 8 – GEN-APP not in open set of generic APPLIDs.
   - C – catastrophic error.
   - 10C – DBLLIST entry ID unknown.
   - 1nn – subtask read codes (timer exit driven, state|messsage file error...).
2. DBLLIST entries have the address of the item specified and the length of the item specified.

Chapter 2.8. Intercommunication Facility Component

This chapter describes the implementation of the CICS intercommunication facility functions using LU6.1 (formerly known as LU6). The intercommunication facility handles communication between CICS and other systems. It covers:

> Communication between CICS systems running in different regions of the same processor, using the interregion communication (IRC) facilities of CICS multiregion operation. Communication is performed by the interregion program (DFHIRP) acting as an SVC, or by the cross-memory program (DFHXMP).
>
> Communication between a CICS system and other systems in the network (ISC), using the facilities of an SNA access method, such as ACF/VTAM, or logical unit type 6 protocols. Other systems can be CICS, or a system, or terminal, that implements a suitable subset of logical unit type 6 protocols.

A comprehensive discussion of this subject using LU6.1, LU6.2 and multiregion operation (MRO) is given in the *CICS/MVS Intercommunication Guide*. The functions described in this chapter are:

- CICS function request shipping
- Transaction routing
- Distributed transaction processing
- Interregion communication.

CICS function request shipping, transaction routing, and distributed transaction processing can be performed using either IRC or ISC.

CICS Function Request Shipping

This section provides an overview of the operation of CICS when it is being used to communicate with other connected CICS systems for CICS function request shipping.

Note: The *CICS/MVS Intercommunication Guide* gives a full description of the **reasons** for CICS function request shipping and how the user can take advantage of the facility.

The CICS function request shipping facility enables separate CICS systems to be connected so that a transaction in one system is able to retrieve data from, send data to, or initiate a transaction in, another CICS system. The facility is available to application programs that use the command-level interface of CICS.

Two or more CICS systems can reside in different address spaces in the same processing system or in different processing systems. When the CICS systems are in different processing systems, data is transferred using VTAM and LU6.1 protocols. When the CICS systems are in different address spaces of the same processing system, then data may also be transferred using VTAM and LU6.1 protocols. It is more efficient, however, if data is transferred using the CICS memory-to-memory SVC provided as part of multiregion operation.

Whether VTAM or the SVC is used to transfer data, the data is represented by LU6.1 function management headers wherever possible.

Application Programming Functions with CICS Function Request Shipping

The functions provided by CICS are extended for CICS function request shipping so that an application program can issue the following types of command and have them executed on another system:

- Temporary storage commands
- Transient data commands
- Interval control commands
- File control commands
- DL/I calls.

Application programs can use these extended functions without being aware of where the resources are actually located; information about where resources are located is contained in the appropriate tables prepared by the system programmer. Alternatively, provision is made for an application program to name a remote system explicitly for a particular request.

Support for sync points, whether explicit (through EXEC CICS SYNCPOINT commands) or implicit (through DL/I TERM calls), allows updates to be made in several systems as part of a single logical unit of work.

Error handling routines may need to be extended to handle additional error codes that may be returned from a remote system. See the *CICS/MVS Intercommunication Guide* for the relevant conditions.

Overview

Local and Remote Names

For a transaction to access a resource (such as a file or transient data destination) in a remote system, it is usually necessary for the local resource table to contain an entry for the remote resource. The name of this entry (that is, the name by which the resource is known in the local system) must be unique within the local system. The entry also contains the identity (SYSIDNT) of the remote system and, optionally, a name by which the resource is known in the remote system. (If this latter value is omitted, it is assumed that the name of the resource in the remote system is the same as the name by which it is known in the local system.)

Mirror Transactions

When a transaction issues a command for a function to be executed on a remote system, the local CICS system encodes the request and sends it to the system identified in the appropriate CICS table. The receipt of this request at the remote system results in the attachment of the CICS-provided mirror transaction CSMI (which uses program DFHMIR). The mirror transaction executes the initiating transaction's request and reflects back to the local system the response code and any control fields and data that are associated with the request. If the execution of the request causes the mirror transaction to abend, then this information is also reflected back to the initiating transaction.

If a resource has browse place holders or is recoverable, the mirror transaction does not free any resources it has acquired until the initiating transaction issues either the appropriate command to free those resources, or a SYNCPOINT or RETURN command.

Provided certain conditions are met, the mirror transaction suspends itself rather than terminates. It is then available to process subsequent requests from MRO-connected systems.

Negotiable Bind for Function Request Shipping (VTAM Only)

Negotiable bind means that a CICS system, when establishing a connection to another processing system, can send its specification of the interface to that system and receive an indication of the capabilities of that system. The result is either an interface acceptable to both systems, or no connection.

You can achieve bind-time security by specifying a **bind password** in the TCT when you define the connection to a remote system. Each pair of communicating systems must have the same password for the link between them. A password is checked each time a session is bound. The checking of a password involves three flows between the systems, as defined in LU6.2 architecture.

When CICS issues OPNDST, a BIND image is sent which defines the widest range of facilities in a

remote system which the CICS system will support. The information sent in the BIND image falls into four classes:

1. Not negotiable – what the primary facility requests must be supported. If a CICS secondary cannot support it, the BIND will be rejected.

2. Negotiable – if the remote system requests this facility, then CICS will adapt to support it (as primary or secondary).

3. Negotiable, but CICS cannot adapt to support this facility – the CICS view should be reflected in a negotiable response by a CICS secondary. A CICS primary will reject a BIND response containing unacceptable settings.

4. CICS does not care whether the remote system has the facility or not.

Initialization of CICS for CICS Function Request Shipping

Provided that CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with ISC=YES (in the SIT or as a console override) causes the following modules to be loaded:

- DFHISP (CICS function request shipping module)
- DFHXFX (optimized data transformation program)
- DFHELR (local/remote decision routine).

The entry point addresses of these modules are contained in the optional features list, which is addressed by CSAOPFLA in the CSA.

The mirror transaction module (DFHMIR) is not loaded until a request is received from a remote system. (DFHMIR can only be loaded if there is a PPT entry for DFHMIR **and** a PCT entry for the mirror transaction CSMI; sample entries are created by the DFHPPT and DFHPCT TYPE=GROUP, FN=ISC table generation macros.)

Note: The ISC=YES option causes other modules besides those specified earlier to be loaded; the ones mentioned here are those specifically required for CICS function request shipping.

Communication with a Remote System

For multiregion operation, communication between CICS systems takes place using module DFHIRP, which runs as an SVC. The SVC moves the data to an intermediate area in key 0 MVS CSA storage, and schedules an SRB to move the data from the intermediate area to the target. For multiregion operation, communication can also be made using the MVS cross-memory services (DFHXMP). Cross-memory services do not require intermediate MVS CSA storage areas.

For ISC, communication between CICS systems takes place via ACF/VTAM links. CICS and the CICS application programmer are independent of, and unaware of, the type of physical connection used by ACF/VTAM to connect the two systems.

Protocols

Requests and replies exchanged between systems for CICS interval control, CICS transient data, CICS temporary storage, and DL/I functions are shipped using the standard protocol as defined for SNA Logical Unit Type 6.1.

Requests and replies for CICS file control functions are shipped using a private protocol (with function management headers of type 43).

CICS Function Request Shipping Environment

System Entries in Terminal Control Table: All remote systems with which a given system is able to communicate are identified and described in terminal control table system entries (TCTSEs); see Figure 79 on page 250. The name of the system entry is the name specified in the SYSIDNT field of the CICS table entry describing a remote resource.

CICS uses the TCTSE as an anchor point to queue requests made by CICS transactions for connection to the remote system.

Figure 80 on page 251 shows three TCTTEs. If a transaction fails, and you get a transaction dump, this figure shows you how to find the relevant TCTTEs from the TCA.

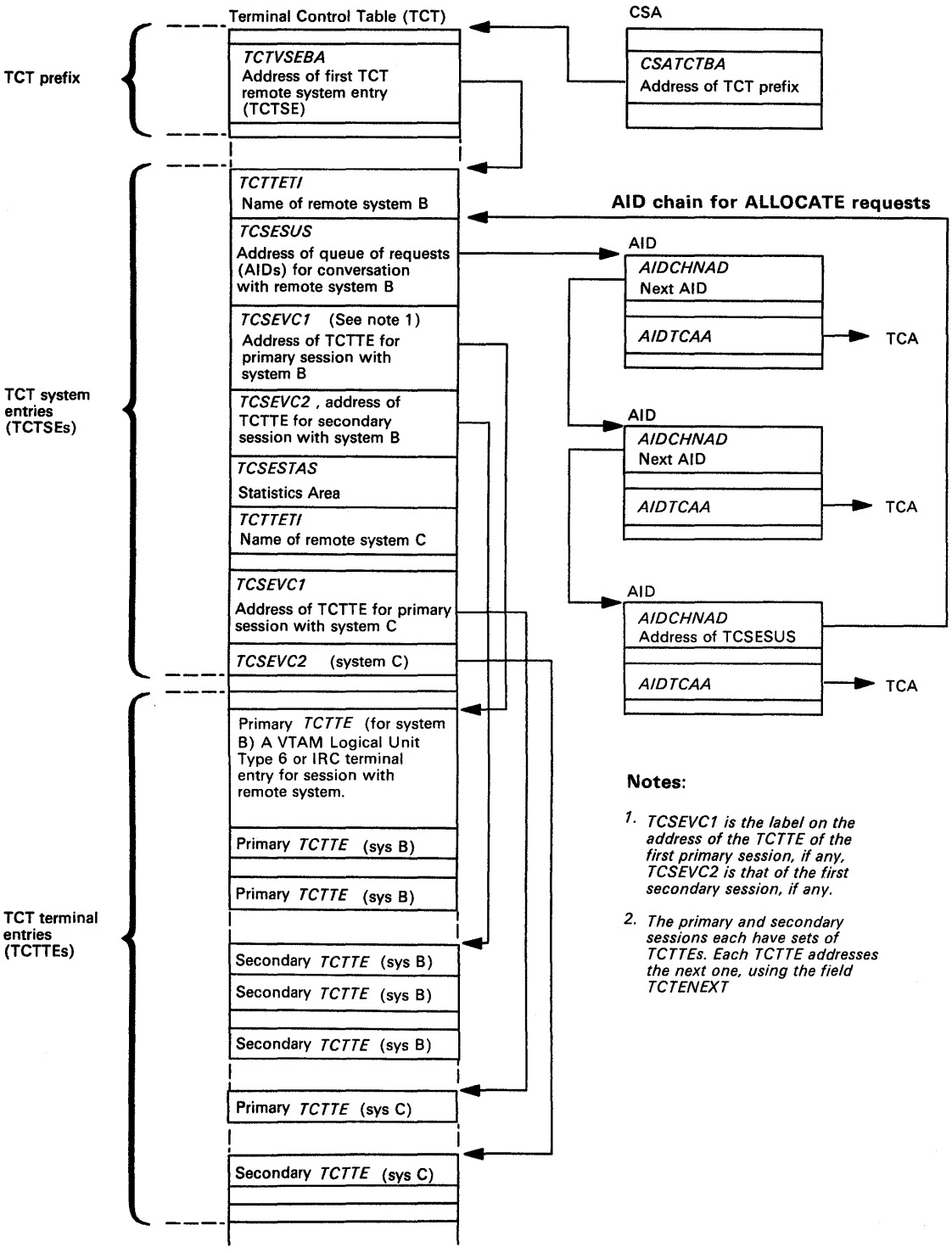


Figure 79. System Entries (TCTSEs) in Terminal Control Table

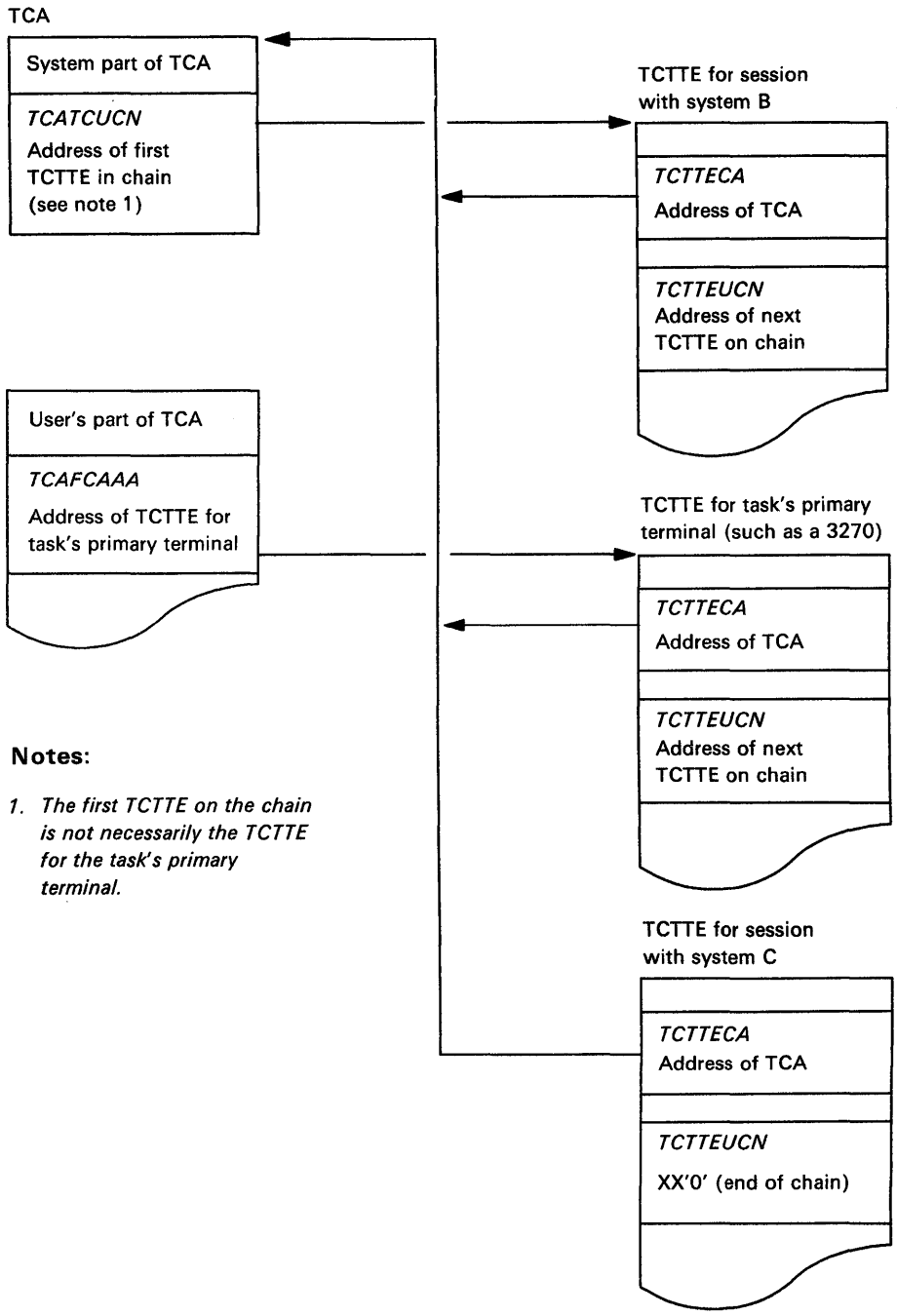


Figure 80. Task's View of CICS Function Request Shipping TCTTEs

Transformation of Requests and Replies for Transmission between Systems

Before a request or reply can be transmitted, it must be transformed from its internal, parameter list (EXEC interface) format to a format suitable for transmission; when received after transmission, the request must then be transformed back into a parameter list format.

There are four such transformations (numbered 1 through 4) and they are all performed by DFHXFX

Transformation 1
For a request to be sent by the originating system; transforms from parameter list format to transmission format.

Transformation 2
For a request received by the mirror transaction; transforms from transmission format to parameter list format.

Transformation 3
For a reply to be sent by the mirror transaction; transforms from parameter list format to transmission format.

Transformation 4
For a reply received by the originating system; transforms from transmission format to parameter list format.

The parameter list format above refers to the parameter list that is normally passed to DFHEIP (for CICS requests) or to DL/I (for DL/I requests).

The transmission formats of these requests and replies (excluding those for sync point protocol) are described in the DSECT DFHFMHDS.

Information that DFHXFX needs to remember between transformations 1 and 4 (in the originating system) or between transformations 2 and 3 (in the mirror system) is stored in a transformer storage area called XFSTG; Figure 81 shows some of the more important fields in XFSTG.

```
XFRSYSNM
Name of remote system
XFRATCTE
Address of session TCTTE
XFRATIOA
Address of current TIOA
XFRPLIST
Address of EXEC parameter list
XFRATABN
Address of entry in resource table
(for example, FCT for file control
resources, DCT for transient data)
XFRFORMN
Data transformation index
(1, 2, 3, or 4 — see text)
```

Figure 81. XFSTG Fields

CICS Function Request Shipping Handling

CICS Function Request Shipping Handling of CICS EXEC Requests

This section describes the sending and receiving of requests and replies (other than DL/I or sync point requests) between two connected systems at the **application-layer** level; see Figure 82 on page 253. (The **function management** and **data flow control** layers, implemented by CICS terminal control, work the same regardless of the type of request being transmitted.)

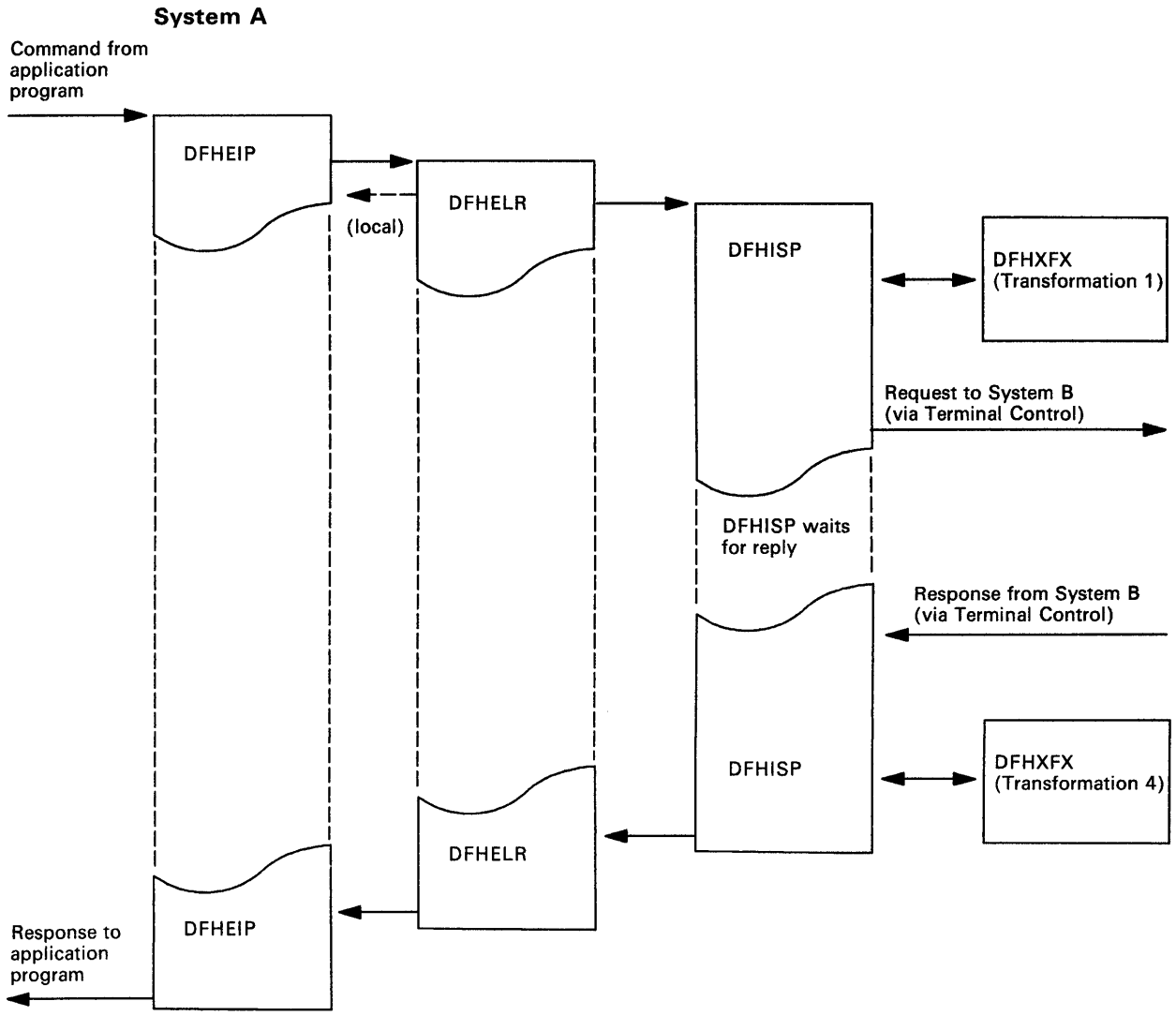


Figure 82 (Part 1 of 2). Overview of CICS Function Request Shipping

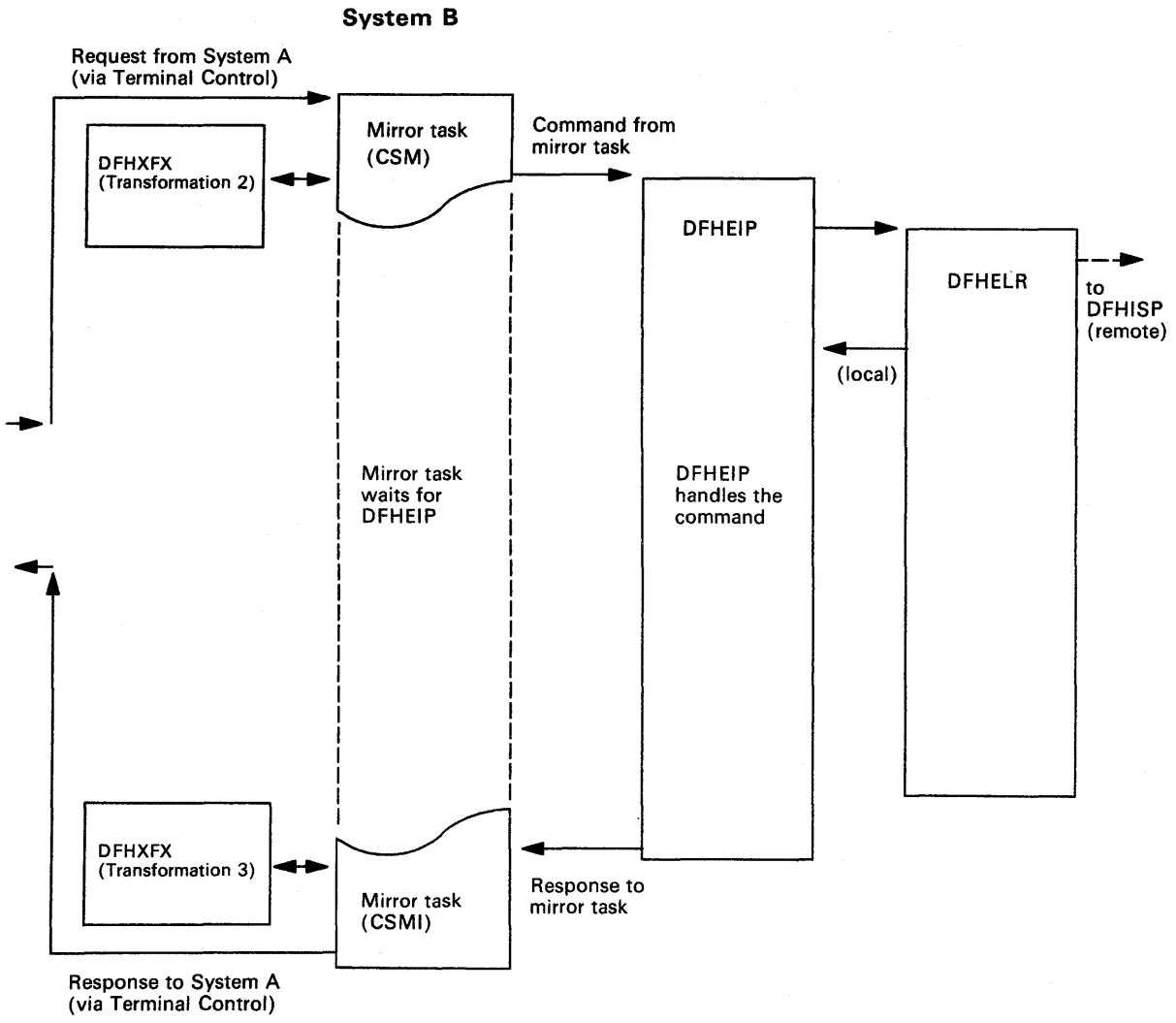


Figure 82 (Part 2 of 2). Overview of CICS Function Request Shipping

Sending a Request to a Remote System: A CICS command is handled for an application program by the EXEC interface program (DFHEIP). DFHEIP analyzes the arguments of each statement to determine the requested function and to assign values into the appropriate CICS control blocks; DFHEIP also performs storage management and error checking on behalf of the application programmer.

If the system has been initialized with ISC = YES in the system initialization table (or as an override parameter), and if the request is for one of the functions that could be executed on a remote system (see "Application Programming Functions with CICS Function Request Shipping" on page 248), DFHEIP invokes a local/remote decision routine called DFHELR which inspects the appropriate CICS table to determine if the request is for a local or a remote resource (unless a remote system has specifically been requested). If the resource is local, DFHELR returns control to DFHEIP which invokes the appropriate CICS modules locally.

Note: A SYSID value which names the local system also causes DFHELR to return control to DFHEIP.

If the resource is remote, DFHELR then:

1. Allocates a transformer storage area (XFSTG) chained off the EXEC interface storage EIS. XFSTG (see Figure 81 on page 252) provides a central area in which all information about processing of the request can be accessed.

2. Places the following data in XFSTG:

   - Name of remote system, for subsequent use by DFHISP (in XFSTG field XFRSYSNM)

   - Address of the application's list of parameters (EXEC parameter list) associated with the command being executed (in XFSTG field XFRPLIST)

   - Address of the table (FCT, DCT, etc.) for the requested resource (in XFSTG field XFRATABN).

3. Issues a DFHIS TYPE=CONVERSE macro which passes control to the CICS function request shipping program DFHISP.

DFHISP obtains the address of the TCTSE for the remote system and places it in XFSTG at field XFRATCSE. DFHISP obtains the address of the TCTTE that controls the session with the remote system and places it in XFSTG at field XFRATCTE. (DFHISP obtains the address by issuing a DFHTC TYPE=POINT macro. If no session is established, there will be no TCTTE; in this case DFHISP issues a DFHTC TYPE=ALLOCATE macro to establish the session TCTTE.)

If no session can be allocated because, for example, all sessions are out of service, then DFHISP determines if the function request can be queued for shipping at a later time. If this proves to be possible, XFRATCTE is set to zero.

Optionally (if a TIOA already exists from an earlier CICS function request shipping request from the same application), DFHISP also places the address of the TIOA in XFSTG (at field XFRATIOA).

DFHISP then invokes DFHXFX to transform the requested command and parameter list into a form suitable for transmission. This is known as **transformation 1** which:

1. Transforms the original **command** into an appropriate type of request for transmission.

2. Converts the EXEC parameter list into a **data unit** having a standardized character-string format (together with a function management header) suitable for transmission. The data unit is built in the TIOA and contains a copy of each of the parameters that are addressed by the EXEC parameter list. (For economy of transmission, certain types of data are compressed before being placed in the TIOA.)

3. Returns control to DFHISP.

4. If local queuing is in effect, then the data unit is built in user storage.

DFHISP then invokes terminal control to transmit the contents of the TIOA to the remote system and waits for the reply from the remote system, if necessary.

If local queuing is in effect, DFHISP issues a DFHIC TYPE=PUT macro specifying transaction CMPX, which will send the data unit at a later time.

Receiving a Request at a Remote System:
Terminal control receives the request transmission and attaches the mirror transaction (CSMI).

The mirror program allocates space for XFSTG; a storage area that is allocated in the LIFO storage area for program DFHMIR. As in the requesting system, XFSTG provides a central area in which all information about the processing of the received request can be accessed. The mirror program places the following data in XFSTG:

- Address of the session TCTTE (in XFSTG field XFRATCTE)

- Address of the TIOA (in XFSTG field XFRATIOA).

The mirror program also allocates scratch pad storage in the LIFO storage area for use by DFHXFX in building argument lists. The address of this storage is placed in XFRPLIST.

The mirror program then invokes DFHXFX to transform the received request into a form suitable

for execution by DFHEIP. This is known as **transformation 2** which:

1. Transforms the received request (as coded in the function management header of the data unit) into an appropriate CICS command.

2. Decodes the TIOA and builds (in the **first** part of the STORAGE area) an EXEC parameter list that basically consists of addresses that point to fields in the TIOA. (Those fields that were compressed for transmission are expanded and placed in the **second** part of the STORAGE area; for these fields, the EXEC parameter list points to the expanded versions, not the compressed versions in the TIOA.)

   Note: The NOHANDLE option is specified on each EXEC CICS command that is created; this has the effect of suppressing DFHEIP's branching to an error routine.

3. Returns control to the mirror program.

The mirror program then invokes DFHEIP (in the same way as for an application program), passing to it (in register 1) the address of the EXEC parameter list just built.

DFHEIP invokes the local/remote decision routine (DFHELR) which determines if the request is for a remote resource on yet another system or for a local resource. If the resource is remote, DFHELR allocates a new and separate transfer storage area XFSTG and invokes DFHISP (as described under "Sending a Request to a Remote System" on page 254.)

If the resource is local, DFHELR returns control to DFHEIP which processes the command for the mirror program in the usual way.

Sending a Reply at a Remote System: The process of sending a reply in response to a request from another system is similar to that for sending a request; see "Sending a Request to a Remote System" on page 254.

When DFHEIP has successfully completed execution of the command, control is returned to the mirror program with the results of the execution in the EXEC interface block (EIB). The mirror program then invokes DFHXFX to transform the command response into a suitable form for the transmission of the reply. This is known as **transformation 3**, which:

1. Checks if the existing TIOA is long enough to take the reply; if not, DFHXFX frees the existing TIOA and creates a new one.

2. Converts the EXEC parameter list (kept in the scratch pad area STORAGE) into a **data unit** having the standardized character-string format suitable for transmission. The data unit is built in the TIOA. If the request is received by the mirror program without CD (that is, the requesting system did not expect a reply), then the mirror program issues a DFHTC TYPE=READ macro, or a DFHTC TYPE=FREE macro. If an error is detected, then the mirror program is forced to abend, so that at least a record of the request failure is written.

3. Returns control to the mirror program.

The mirror program then invokes terminal control to transmit the TIOA. (The mirror program does this by issuing a DFHTC TYPE=(WRITE,WAIT,READ) macro if the mirror program holds any state information that must be held for a further request or until a sync point. Otherwise, a DFHTC TYPE=(WRITE,LAST) macro is issued; see "Mirror Termination Logic (MRO Only)" on page 265.)

RECEIVING A REPLY FROM REMOTE SYSTEM: Terminal control receives the reply and returns control to the initiating task; in particular, control is passed to DFHISP, which has been waiting for the reply.

DFHISP invokes DFHXFX (passing to it the address of the XFSTG area) in order to transform the reply into the form expected by the application program. This is known as **transformation 4** which:

1. Obtains the addresses of the TIOA and of the original EXEC parameter list from XFRATIOA and XFRPLIST in the XFSTG area.

2. Uses data in the reply to complete the execution of the original command. For example:

- Sets return codes in the EIB from status bits in the reply.
- Stores other received data (if any) in locations specified in the original EXEC parameter list.

3. Frees the TIOA.
4. Returns control to DFHISP.

DFHISP returns control through DFHELR to DFHEIP which raises any error conditions associated with return codes set in the EIB. DFHEIP then returns control to the application program.

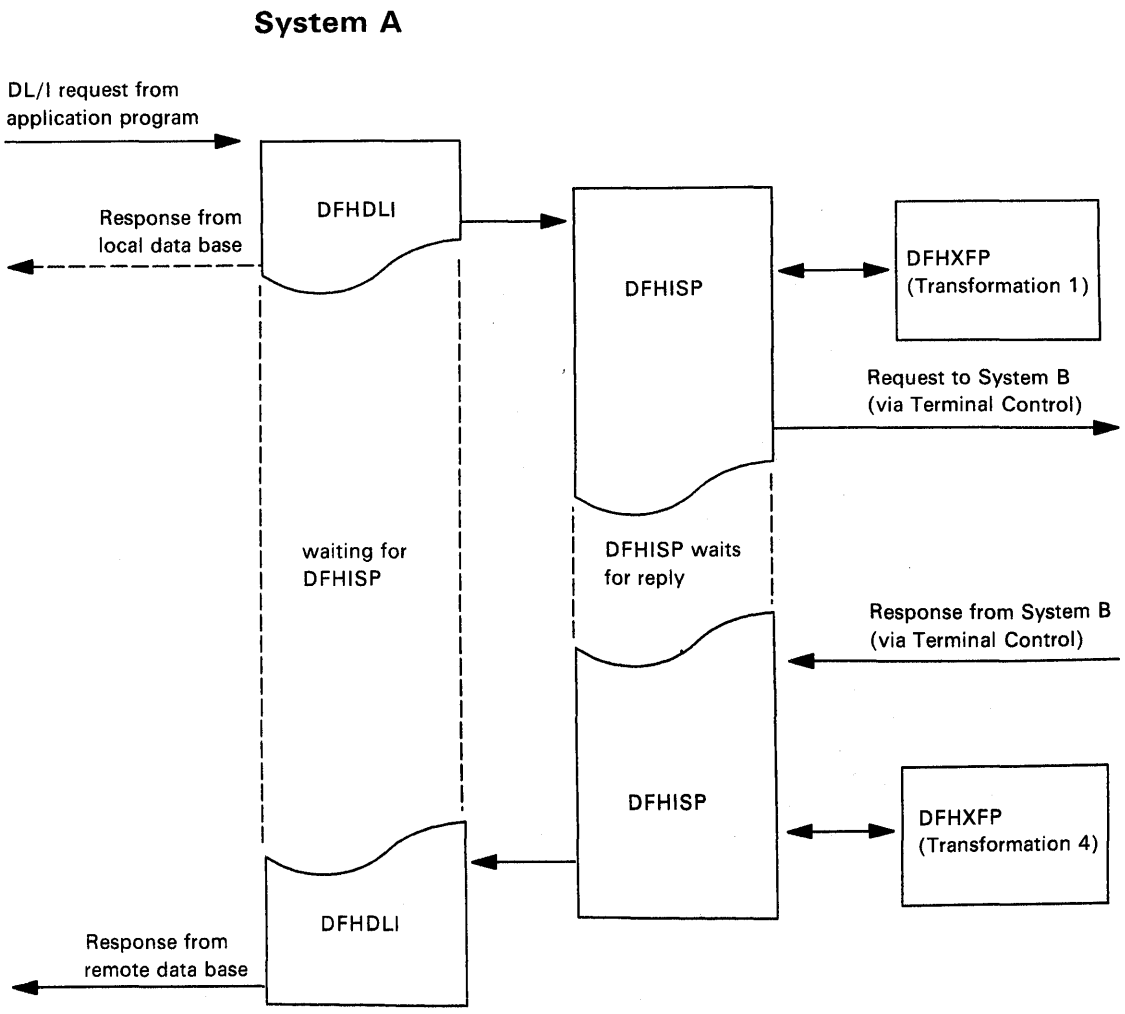


Figure 83 (Part 1 of 2). Overview of CICS Function Request Shipping Handling of DL/I Requests

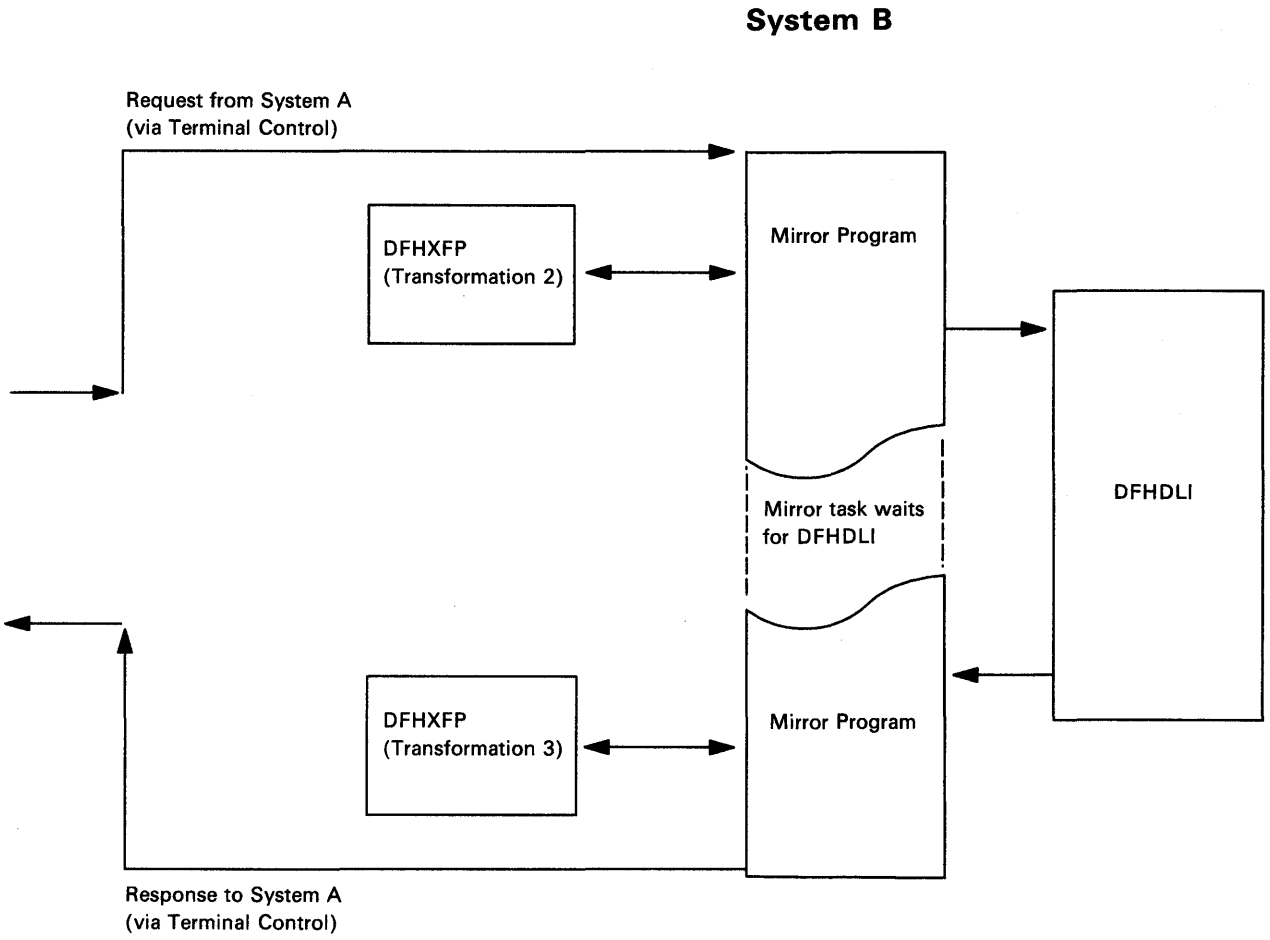


Figure 83 (Part 2 of 2). Overview of CICS Function Request Shipping Handling of DL/I Requests

CICS Function Request Shipping Handling of DL/I Requests

DL/I requests are handled in a similar manner to that for CICS commands; see Figure 83 on page 257.

SENDING A DL/I REQUEST TO A REMOTE SYSTEM: Whereas all CICS commands (for a local or remote system) are handled by DFHEIP, all DL/I requests are handled by DFHDLI.

DFHDLI checks whether the request is for a local or remote data base. If local, the request is handled normally. If remote, control is passed to DFHISP by using the DFHIS TYPE=CONVERSE macro.

DFHISP then:

1. Invokes DFHXFP to transform the request into a form suitable for transmission.
2. Invokes terminal control to transmit.

RECEIVING A DL/I REQUEST AT A REMOTE SYSTEM: As for a CICS request, the mirror transaction (CSMI) is attached and then invokes DFHXFP to transform the received request into a form suitable for execution by DFHDLI.

The mirror program then passes the request to DFHDLI in the same way as any other application program would. (As distinct from the way in which CICS commands are treated, DFHDLI in the receiving system makes no attempt to check if the request is for yet another remote system.)

SENDING A DL/I REPLY AT A REMOTE SYSTEM: When DFHDLI has successfully completed the request, control is returned to the mirror program with the results in the user interface block (UIB). The mirror program then:

1. Invokes DFHXFP to transform the results into a form suitable for transmission.

2. Invokes terminal control to transmit the reply.

RECEIVING A DL/I REPLY FROM A REMOTE SYSTEM: On receipt of the reply, terminal control returns control to DFHISP, which has been waiting for the reply; DFHISP then invokes DFHXFP to transform the reply into a form that can be used by DFHDLI. DFHXFP sets the return codes in an intermediate control block, DFHDRX, so that they can ultimately be copied to the UIB or the TCA for the application program. Control is then returned through DFHISP and DFHDLI to the application program.

CICS Function Request Shipping Handling of Sync Point Requests

When the application takes a sync point (either a user sync point or a RETURN), then the requirement for a sync point is shipped to all mirrors that are attached (and this includes those that have updated a recoverable resource since the last sync point). The mirrors then take their own sync point and terminate. Thus, changes made to local and remote resources are all committed together, or in the event of a failure, are backed out together.

The simplest case of synchronizing the commitment of changes to local and remote resources is between an application and a single mirror; see Figure 84 on page 260.

Note: The shipping of sync point functions to a remote system is handled by the sync point program, DFHSPP – not DFHEIP.

SYNC POINTING WITH A SINGLE MIRROR TASK: The application program requests a remote update to a recoverable resource. This attaches the mirror which performs the update, replies, and suspends waiting for the next request. When the application takes a sync point, it first **prepares** its own resources for commitment; that is, it flushes out buffers and logs its updates to tables such as the destination control table and the temporary storage unit table. This is done by the sync point program in its first-pass deferred work element (DWE) processing and DWE logging pass as in a non-CICS function request shipping environment. If anything goes wrong at this stage, the application abends and backs out. An ABORT command is shipped to the mirror which then backs out also.

Having prepared its own resources, the application sends a sync point request (SPR) to the mirror, and suspends while waiting for a positive response. Should the response be negative because something went wrong in the mirror (for example, system log failure), the application abends and backs out, thus keeping the local and remote resources in synchronism.

When the mirror system receives the sync point request, it enters the sync point program which prepares its resources and then **commits,** that is, it writes end-of-task to the system log, so that even in the event of a system failure, the changes will not be backed out during any subsequent emergency restart. The mirror program then (1) sends the positive response back to the application, (2) dequeues to release its exclusive control of the recoverable resources, and (3) terminates.

When the application receives the positive response, it too commits and dequeues.

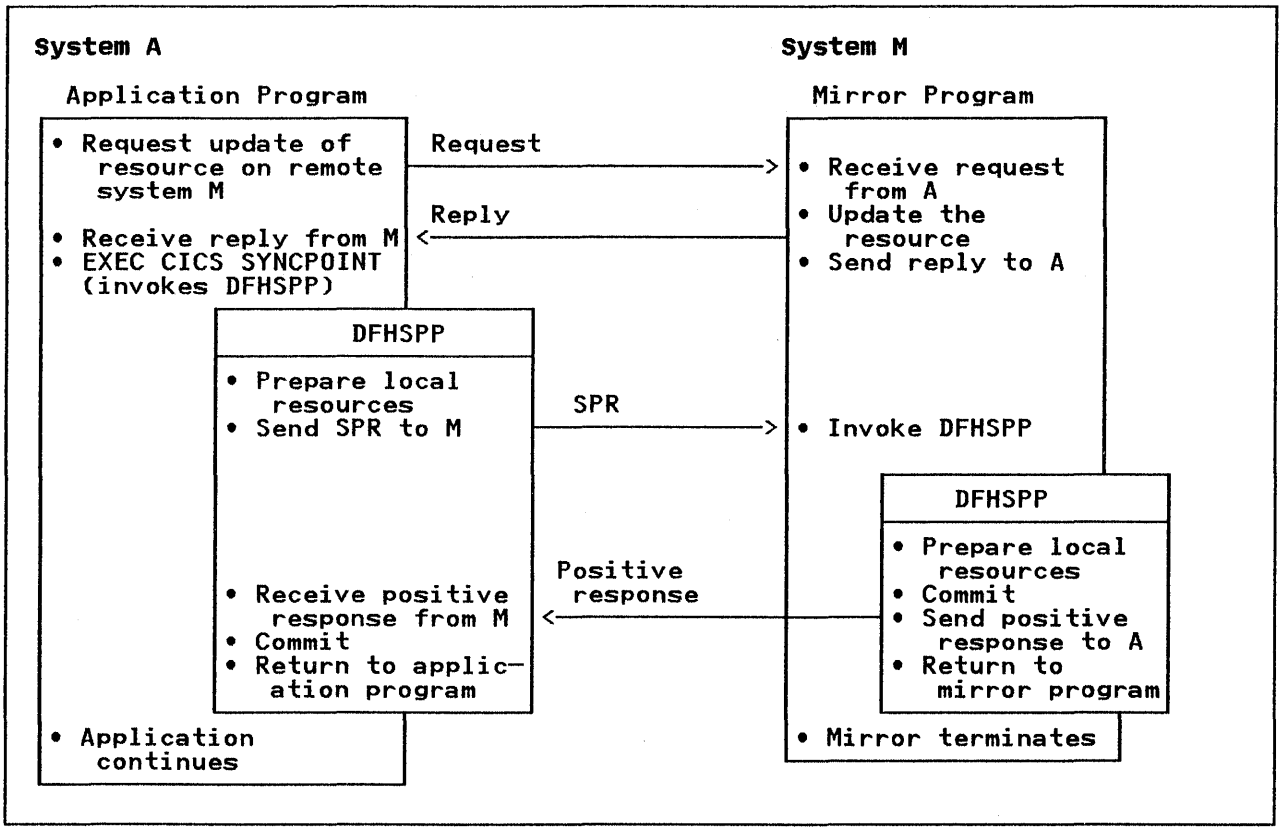


Figure 84. Sync Pointing with a Single Remote System

SYNC POINTING WITH MULTIPLE MIRROR TASKS: When more than one mirror is active at the time of the sync point, the simple flow of SPR and response between the application and mirror is insufficient. (For example, if the first mirror successfully commits but the second mirror fails, it would be too late to back out the first mirror's commitment and the resources would be left out of step.) This situation is avoided by using a sync point function called **PREPARE**, which instructs a mirror to prepare its resources (for commitment) and then reply with an SPR. Having prepared its resources, the mirror is in a state where it can back out or commit according to the response to the SPR. Thus, the flow between an application with two mirrors M1 and M2 would be as shown in Figure 85 on page 261. If anything goes wrong in either of the mirror systems prior to **commit**, then all the systems can be backed out by sending ABORT or negative response to the other systems.

Note: For the last (or only) mirror addressed by the application during sync point processing, an SPR is sent; but for other mirrors, a PREPARE is sent (and an SPR is awaited in return). Thus, in Figure 85 on page 261, the flow between the application and mirror M2 (the **last** mirror) is the same as that for the simple case of a single mirror shown in Figure 84 on page 260.

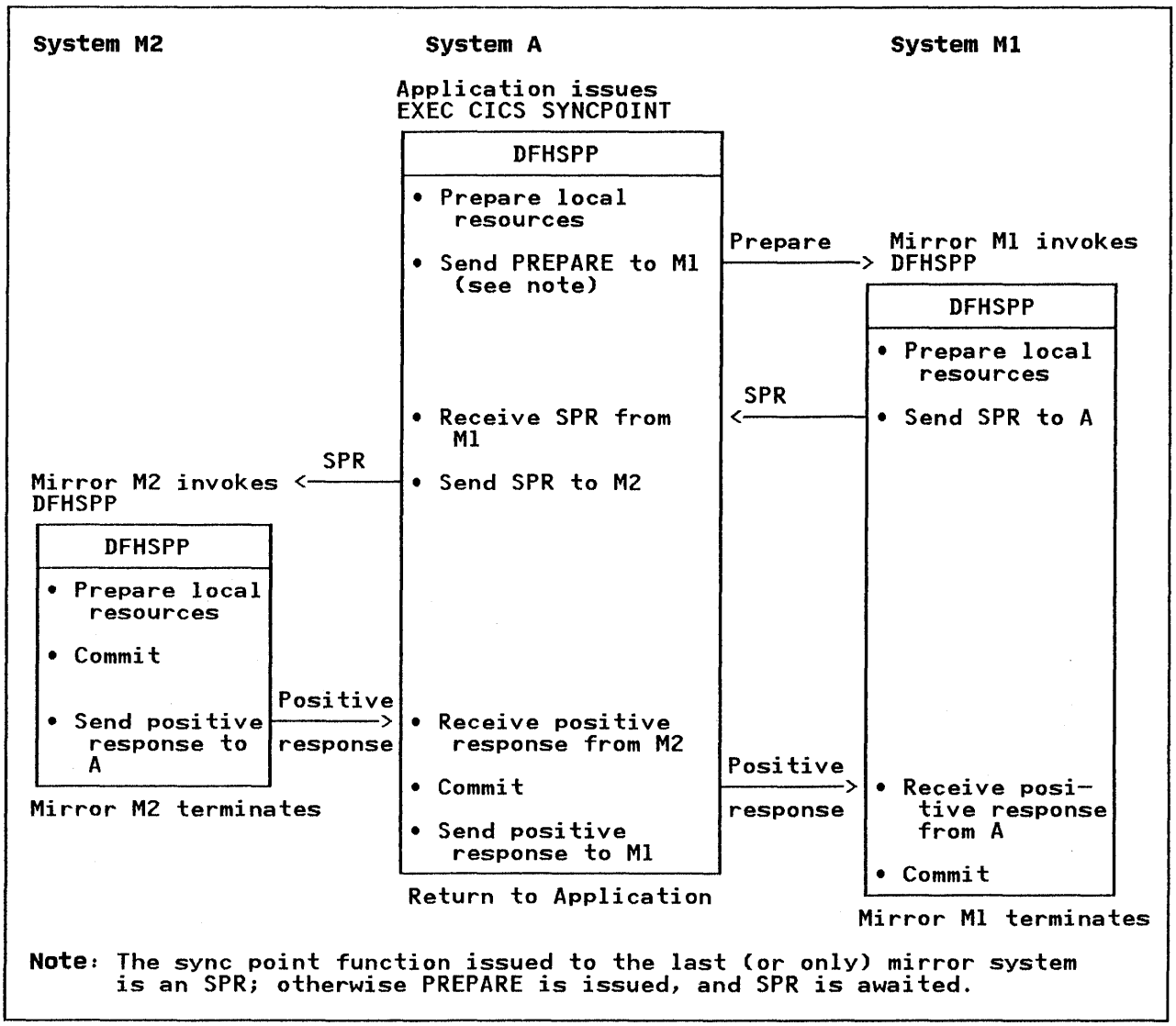


Figure 85. Sync Pointing with Multiple Remote Systems

The three sync point functions (PREPARE, SPR, and positive response to SPR) are sent by DFHSPP using the DFHTC CTYPE=PREPARE, CTYPE=SPR or CTYPE=COMMIT macros; see "Terminal Control Support for CICS Function Request Shipping" on page 265. Should any failure occur before commit time, then DFHSPP sends ABORT to all mirror systems using DFHTC CTYPE=ABORT.

If a mirror has further mirrors attached to it (by **daisy chaining**), then as part of its own **prepare-local-resources** function, the mirror behaves as an application program with respect to its own mirrors. That is, the mirror completes the PREPARE, SPR, and response sequence (or just SPR and response) with its daisy chained mirrors before responding to the application's original sync point function; see Figure 86 on page 263.

Note: The flow between the mirror M1 and its two mirror systems, M2 and M3, is the same as that shown in Figure 85 for an **application** and its two mirror systems.

CICS Function Request Shipping Sync Points and Session Failures: From the foregoing text, it can be seen that, excluding session or system failures (as opposed to transaction failures), DFHSPP can ensure that local and remote updates to recoverable resources are either all committed or all backed out (assuming dynamic backout is in effect). The effect of a session failure between an application and its mirror is now considered.

Session failures **before** sync point processing has been entered present no special problem because the application at one end or the mirror at the other will abend and back out, thus leaving resources in synchronization.

Session failures **during** sync point processing can mean that one side does not know whether the other side has committed or backed out. For example, suppose that the application has just sent SPR and is suspended waiting for the reply when the session fails. The mirror may not have received the SPR and will have backed out because of the failure; alternatively, the mirror may have received the SPR, committed, and sent a positive response which never reaches the application.

In special circumstances described later, the suspended system waits until the session is recovered, compares message sequence numbers to establish whether the other system committed or not, and then follows suit. However, because of the complications involved in keeping the affected resources locked against further changes until session recovery, CICS in general decides unilaterally whether or not to complete; where this means that changes to recoverable resources may be out of sync, CICS issues messages to warn the user.

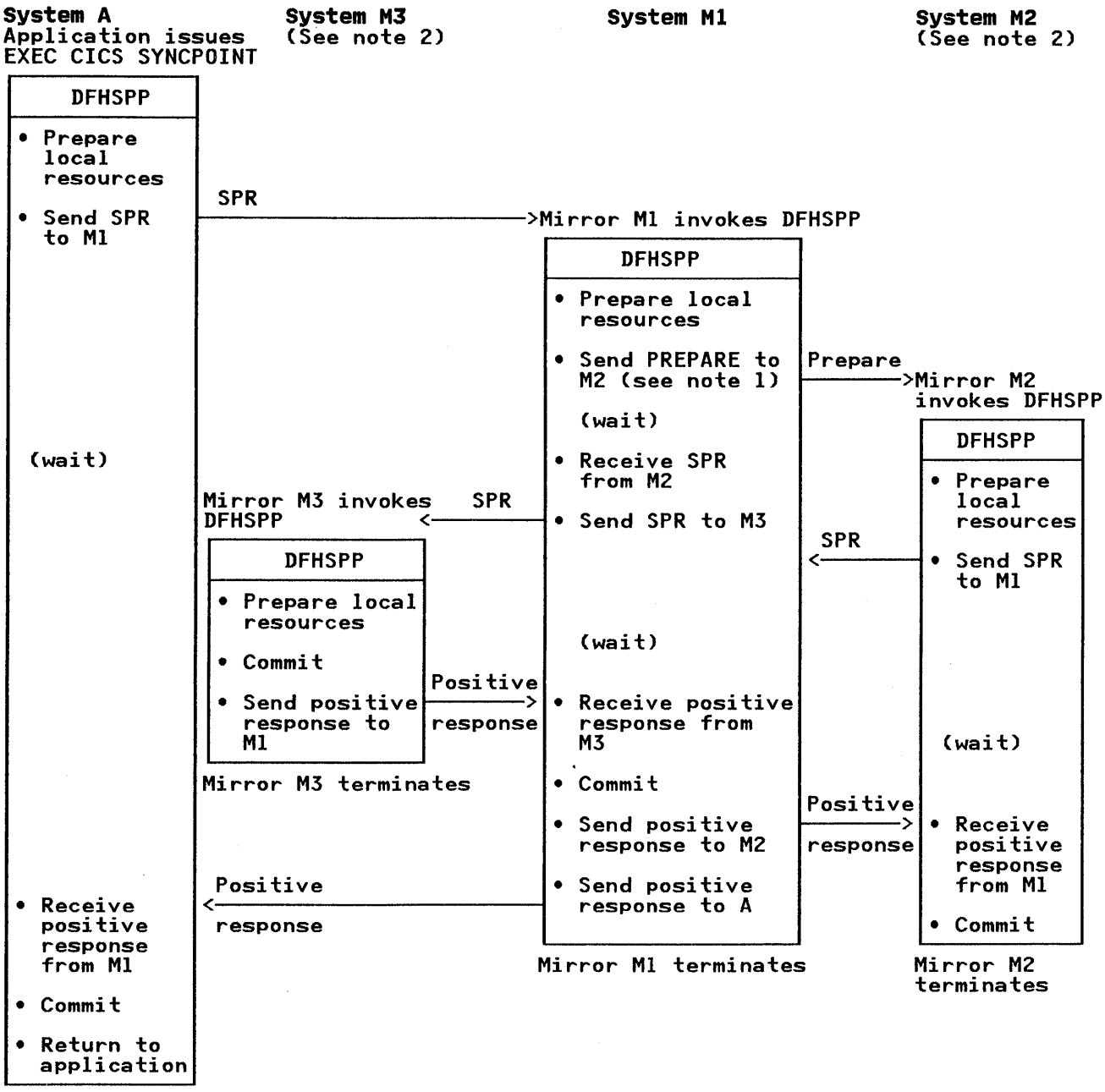
The rules governing the unilateral decision are as follows:

1. A mirror completes and commits if a session fails at such a time that the application might have completed – for example, when the mirror sent SPR in reply to PREPARE and the session failed before the mirror received the response. Because the application might have completed and gone away with no indication of an error, it is important that the mirror also completes.

2. The application abends if a session fails while there is doubt about whether a mirror has abended or completed, that is **after** the application sent SPR or PREPARE but **before** the response (or SPR reply) was received. (Dynamic backout of the application transaction may however be suppressed – see the *CICS/MVS Intercommunication Guide*.) Whenever the application abends while in this **in-doubt** state, message DFH2101 (warning of possible loss of synchronization of data base changes) is sent to CSMT in the application local system. This message has several inserts, for example, time of failure and task number, for cross referencing with the DFH2102 or DFH2103 messages described below.

As mentioned before, CICS waits across session failures during sync point without committing or backing out under special conditions which are:

- Only one remote system is connected to the TCA at the time and it is ISC and not the multiregion option.
- DTB = (YES, WAIT) is specified in the PCT for the transaction.
- The only recoverable changes made during the unit of work are EXEC CICS WRITEQ TS or EXEC CICS START PROTECT commands.



Notes:

1. The sync point function issued to last (or only) mirror system is an SPR; otherwise PREPARE is issued, and SPR is awaited.
2. In this example, mirror systems M2 and M3 are each daisy chained from M1.

Figure 86. Sync Pointing with Daisy Chained Mirror Systems

The **wait state** is achieved by transferring the deferred work elements (DWEs), representing the commit or backout work to be done, from the TCA to the TCTTE of the failing session. The lock ownership is also transferred. The task then completes normally in other respects (with message DFH2105). If you are using ISC, DFHZNAC is invoked on session recovery. DFHZNAC transfers the DWEs and locks from the TCTTE to its own TCA and executes a sync point with or without ROLLBACK, as appropriate, to keep in step with the remote system.

Note: The contents of the DWEs are logged and, following system failure, the DWEs are reconstructed by emergency restart.

When the session is eventually recovered, SNA sequence numbers are compared by DFHZRSY (for ISC) or DFHCRR (for multiregion operation) to see whether the two systems are still in sync or not. If they are, then message DFH2102 is issued; otherwise DFH2103 is issued. Both messages carry the same inserts as the original DFH2101 message issued when the session failed.

The session numbers to be compared on session recovery are maintained by DFHSPP in the TCTTE for the session, (in fields TCTESQIL and TCTESQOL). They correspond to the sequence numbers of the last successfully processed inbound and outbound SPR requests.

The message insert information is remembered from session failure to session recovery in a control block built by DFHSPP which is pointed to by the TCTTE field TCTEMII. DSECT DFHIMSDS describes the control block.

In order to allow proper resynchronization during session recovery when parallel sessions are being used, it is necessary for both sides to know which parallel session is being resynchronized.

A remote system is specified by NETNAME in the TCTSE. A particular session with a remote system is known to CICS by the TCTTE name. The remote system has an equivalent name for the session which CICS needs to know when the remote system has particular resources associated with the session; for example sequence numbers at session recovery, or a message queue. For intersystem communication, the name can be made known to CICS either at TCTTE generation by use of the NETNAMEQ operand, or at session initiation in the REQSESS, BIND, and RESPONSE exchanges. The remote system's name for the session is remembered in the node initialization block (NIB) descriptor for session recovery. For multiregion operation, the name is made known during session recovery by means of the transaction CSIR (program DFHCRR).

The two names together are known as the session qualifier pair (SQP) and fully identify the session.

DFHSPP and DFHWKP log the remote session qualifier (from the NIB descriptor), along with the sequence numbers, in the SPR journal record which is written during sync point. Thus, when sequence numbers are recoverable, so are session qualifiers.

During emergency restart, DFHRUP recovers the sequence numbers and the remote session qualifier and restores the latter to the TCTTE's NIB descriptor. Consequently, the full SQP is known for session recovery.

CICS Function Request Shipping Sync Points and System Failures: A CICS system failure appears to the running system at the other end of the session as a session failure and is treated as described above. The failing end is restarted using emergency restart. This has the effect of restoring the system to the state that it would have been in if just the session had failed, and not the whole system. Thus the application transaction and its mirror are backed out or not by the same decision rules, and message DFH2101 or DFH2105 is issued as if the session had failed while the application was **in-doubt**. When the session is restored, messages DFH2102 or DFH2103 are issued just as for simple session failure and recovery.

In order that the above may be achieved, the following information is recorded on the system log for the correct status to be restored by emergency restart:

1. The SPR sequence numbers described above
2. Whether the application is currently **in-doubt** about whether the mirror backed out or completed
3. The message inserts, required if the system failed when **in-doubt**.

The information is contained in an SPR log record (described by the DFHJCA or DFHJCR macro). The record is written by DFHSPP when an SPR record is about to be written or when one is expected to be received in response to PREPARE. It contains the appropriate sequence numbers assuming the sync point completes normally. The normal completion of the sync point is indicated by a following logical-end-of-task or physical-end-of-task record on the log.

If the transaction abends during sync point processing and dynamic backout is applicable, then the sync point records (DCT, TSUT etc.) logged so far are effectively canceled as follows:

1. The end-of-task record is not written.

2. Dynamic backout program is invoked and then the sync point program is reinvoked. (This causes a new start-of-sync-point log record followed by the records for the backed out versions of the DCT and TSUT.)

3. An end-of-task record is then written at the end of this attempt at the sync point.

The SPR records contain not only the sequence numbers, but also the message insert information in case it is needed following emergency restart. In addition, the SPR records that are written before sending a PREPARE or an SPR not in reply to PREPARE, contain an **in-doubt** flag. This flag indicates that if the system fails before end of task, then the application is **in-doubt** as to whether the mirror completed or backed out.

The **in-doubt** period ends at normal sync point completion, at successful transmission of an ABORT command, or on session recovery following an **in-doubt** session failure. The last two events must therefore also be logged in addition to the first one. The successful transmission of ABORT is logged by DFHSPP in an SPR record with an ABORT flag set. The session recovery case is logged by DFHZNAC in an ordinary **not-in-doubt** SPR record.

There is one more situation to be covered. If a session fails while **in-doubt** and DTB=(YES,NO) has been specified in the PCT, then the transaction is not to be backed out (see the *CICS/MVS Intercommunication Guide*). Therefore a normal end-of-task record is required to commit the normal sync point logging. However, the in-doubt period has not ended. This is indicated by DFHSPP logging a special SPR record with a **session-failed** flag set.

The activity keypoints written periodically to the system log also contain condensed copies of the current state of session TCTTEs (described by DFHKPTE) to avoid an indefinitely long backward scan of the system log during Emergency Restart. This activity keypointing is done by DFHAKP as usual.

During emergency restart, DFHRUP gathers the relevant information from SPR records and their context (and from activity keypoints) on the system log and saves it in the message backout table, (DFHMBODS) as is done for TCTTEs that are not used for sessions.

The message backout table is passed to DFHTCBP which restores the TCTTE to the state it would have been in had the system failure been just a session failure. It issues message DFH2101 or DFH2105 if the session was **in-doubt** and chains the message insert information off the TCTTE ready for the DFH2102, DFH2103, DFH2106 or DFH2107 messages on eventual session recovery.

Mirror Termination Logic (MRO Only)

The mirror transaction suspends itself after issuing a sync point and freeing the session to which it is attached. It is then available to process subsequent MRO requests.

Terminal Control Support for CICS Function Request Shipping

Terminal control support for CICS function request shipping falls into the following main areas:

1. TCTTE allocation functions, ALLOCATE, POINT, and FREE. These functions are used mainly by DFHISP to allow a CICS transaction to own additional TCTTEs. These are session TCTTEs to remote systems; these functions are supported by DFHZISP.

2. Sync point functions, SPR, COMMIT, ABORT, and PREPARE. These functions are used by the sync point program DFHSPP to

implement the sync point protocol; these functions are supported by DFHZIS1.

3. logical unit type 6.1 functions. These functions are used by users of terminal control to support the data flow control protocols used in a logical unit type 6.1 session.

The functions described in notes 1 and 2 on page 265 are extensions to the DFHTC macro which are intended for internal use by CICS control programs only; they are not documented in the user manuals.

TCTTE Allocation Functions

Terminal control provides the following TCTTE-related functions:

ALLOCATE function
This allocates to the requesting transaction a session TCTTE for communication with a remote system. The name of the remote system is passed as a parameter. The address of the allocated TCTTE and/or a return code is returned to the requestor. DFHZISP uses the DFHZCP ATI (automatic transaction initiation) mechanism to allocate the session.

If the allocation request cannot be satisfied immediately, an AID (automatic initiate descriptor) is created, and is chained off the system entry; the AID is used to remember, and subsequently to process, the outstanding allocate request.

Parallel sessions can be allocated explicitly, or implicitly by reference to a remote resource; sessions are automatically initiated at allocation time, if necessary. They can also be initiated by a master terminal ACQUIRE command, or automatically during CICS initialization if CONNECT=AUTO is specified in the TCTTE.

POINT function
This causes terminal control to supply the requesting task with the address of a session TCTTE for a named remote system. The TCTTE must have been previously allocated to the requesting task.

FREE function
This detaches a TCTTE from the owning task and makes it available for allocation to another transaction. (The FREE function is the opposite of the ALLOCATE function.)

TERM=YES operand
This operand enables the issuer of a terminal control macro to select explicitly the TCTTE to which the requested function will be applied. The address of the TCTTE to be processed is passed as a parameter of the request; the TCTTE must have been previously allocated to the requesting task.

FREE TCTTE indicator
This indicator is set as a result of the remote system issuing a (WRITE,LAST) or FREE request to show that the current conversation has finished and that the session should be freed by the current owner of the TCTTE. The receiver of the FREE indicator (usually DFHISP) must issue a FREE request.

Sync Point Functions

For ISC, terminal control provides the following sync point functions (the equivalent functions for IRC are provided by DFHZIS1):

SPR (sync point request) function
This request is issued by DFHSPP during sync point processing, and causes terminal control (DFHZSDR) to send a request that has a definite DR2 response requested. This tells the other side of the session that a sync point is required.

COMMIT function
This request is issued by DFHSPP when sync point has been completed. It causes a positive DR2 response to be sent, signaling the successful completion of sync point protocol.

ABORT function
This request causes either a negative DR2 response or a LUSTATUS command to be sent, indicating that a requested sync point operation could not be completed successfully, or that there has been an abnormal end of the current logical unit of work.

PREPARE function
This request causes an LUSTATUS command to be sent to the mirror in the remote system and indicates that a sync point should be taken.

Logical Unit Type 6.1 Protocols for ISC

CICS acts as both the primary and secondary half of an LUTYPE6 session. To implement secondary half-session support, CICS VTAM terminal control has to do two things:

Implement the secondary half of the data flow control and session control protocols that CICS already uses as a primary.

Use the secondary API provided by VTAM.

The terminal control functions provided by CICS are independent of primary/secondary considerations. Differences between the primary and secondary VTAM interfaces are contained within the CICS modules that issue the appropriate VTAM request. The secondary support functions appear principally in the DFHZCP modules shown in Figure 87.

Module	Function	Secondary Function
DFHZSIM	Request LOGON	Use REQSESS macro
DFHZOPN	OPNDEST	Use OPNSEC macro
DFHZSCX	SCIP exit	Receive and process BIND, STSN, SDT, CLEAR, and UNBIND commands
DFHZCLS	CLSDST	Use TERMSESS macro
DFHZRSY	Resynchronization	Build STSN responses
DFHZSKR	Respond to	Send responses to BIND, SDT, and STSN commands
DFHZRAC)		
DFHZRVX)	Receive	Receive and process BID commands
DFHZATI)		
DFHZRVX)	Bracket protocol	Implement secondary contention resolution using bracket protocol
DFHZRAC)		
DFHZNSP	Network services error exit	Handle secondary LOSTERM type of errors

Figure 87. VTAM Secondary Support Functions

SYMMETRICAL BRACKET PROTOCOL: Logical unit type 6.1 sessions between two CICS systems require most protocols to be symmetrical; therefore, CICS receives (as well as sends) end bracket.

SHUTDOWN PROTOCOL: The logical unit type 6.1 shutdown protocol does not use the SHUTDOWN command.

The logical unit type 6.1 shutdown protocol uses the data flow control commands SBI (stop bracket initiation) and BIS (bracket initial stopped). Shutdown is executed as part of session termination (by DFHZCLS) and ensures that, when a session is terminated normally (as a result of a master terminal release command or a normal CICS shutdown), there are no unfinished sync point requests on the session. This means that when the session is initiated, no resynchronization sequence is required.

SENDER ERP (ERROR RECOVERY PROTOCOL): CICS support for logical unit type 6.1 uses a symmetrical SNA protocol called **Sender ERP**. In addition, when CICS wishes to send a negative response to a remote system, it sends a special negative response (0846), which indicates that an ERP message is to follow. This ERP message contains the real system and user sense values, together with a text message. The negative response and ERP message are built by DFHZEMW and are received and processed by DFHZRPX, DFHZRAC, DFHZRVX, and DFHZNAC.

RESYNCHRONIZATION PROTOCOL: CICS support for logical unit type 6.1 sessions that use the sync point protocol described earlier in this

chapter has associated resynchronization logic, which is used during the initiation of a session after a previous session has terminated abnormally. This logic is used to generate messages concerning the outcome of any logical units of work that were **in-doubt** when the previous session failed. The modules involved are DFHZRSY, DFHZSCX and DFHZNAC.

CICS Function Request Shipping NOCHECK Option for ISC

The NOCHECK option significantly reduces the utilization of the processors and of the communication link; this is achieved by reducing the CICS-generated control sequences that are normally exchanged.

The NOCHECK option is always required if the request is being shipped to IMS/VS.

NOCHECK Option Function Handling

The transmission of a START NOCHECK command and associated data is handled in a slightly different manner from that for other CICS function request shipping commands. Compared with the process described earlier in this chapter in "CICS Function Request Shipping Handling of CICS EXEC Requests" on page 252, the major differences are:

- After DFHISP has allocated the session TCTTE to the requesting task, the transformer program DFHXFX performs **transformation 1**. In addition, DFHXFX detects that a START NOCHECK command is being processed and passes this fact to DFHISP in its return code. Accordingly, DFHISP issues a DFHTC TYPE=WRITE macro which is deferred until sync point, return or another ISC session.
- DFHISP returns to its caller.
- On the receiving system, DFHEIP handles the START NOCHECK command in the usual way and then terminates when the command has been executed; no response is sent back to the first system.

Compatibility

Compatibility at the CICS command-level interface to application programs will be maintained for all CICS function request shipping facilities in any future extension or enhancement.

Compatibility at the CICS-implementation level will be maintained for remote access to transient data, temporary storage, or DL/I data bases, but not necessarily for access to remote files.

Transaction Routing

This section provides an overview of the operation of CICS when it is being used to communicate with other connected CICS systems for transaction routing.

Note: The *CICS/MVS Intercommunication Guide* gives a full description of the reasons for transaction routing, and how the user can take advantage of this facility.

The transaction routing facility enables a terminal operator to enter a CICS transaction code into a terminal which is attached to one CICS system and thereby start a transaction on another CICS system which is in a different address space in the same processing system, or another system. The concept of local and remote names is the same.

Application Programming Functions with Transaction Routing

The transaction that is initiated from a terminal connected to another CICS system by means of the transaction routing facility can execute using programs written in either the macro-level interface or the command-level interface. The application program can issue terminal control requests directly, or can use the BMS or batch data interchange functions. The transaction can be initiated by data entered from a terminal, or by automatic transaction initiation. The application program can use these functions without being aware of where its terminal is connected to (or in the case of BMS routing, where the terminals in the route list are connected to). Information about where terminals and transactions are located is

contained in the appropriate tables prepared by the system programmer.

Overview of Transaction Routing

When a terminal operator enters a transaction code for a transaction which is in a remote address space, the transaction that is attached executes the CICS relay program DFHCRP. The transaction is referred to as the relay transaction, although it is in fact a user-defined transaction with user-defined attributes, executing the CICS relay program. The relay transaction sends a request to the remote address space which causes the user transaction to be started in that address space. When the user transaction issues a request to its principal terminal, that request is intercepted by the CICS terminal control program, shipped back to the relay transaction, and subsequently to the real terminal.

The relay transaction is active while the user transaction is executing. When the user transaction terminates, it sends a message for the relay transaction to terminate, making the terminal available to other users. When the user transaction takes a sync point, the relay transaction is informed, if necessary, in which case it takes a sync point.

Negotiable Bind for Transaction Routing

Negotiable bind for tranaction routing is the same as that for function requesting shipping, see "Negotiable Bind for Function Request Shipping (VTAM Only)" on page 248 for details.

Initialization of CICS for Transaction Routing

Provided that CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with ISC=YES (in the SIT, or as a console override) causes the following modules to be loaded:

- DFHXTP (transaction routing data transformation program)
- DFHZCX (which includes the CSECT DFHZTSP, the terminal sharing program).

The entry point addresses of these modules are contained in the optional features list that is addressed by CSAOPFLA in the CSA.

The CICS relay program, DFHCRP, is not loaded until a **remote** transaction is initiated. DFHCRP can only be loaded if there is a PPT entry for DFHCRP; a sample entry is created by the DFHPPT TYPE=GROUP, FN=ISC table generation macro.

Note: The ISC=YES option causes other modules besides those specified earlier to be loaded; the ones mentioned here are those specifically required for transaction routing.

Transaction Routing Environment

The connection environment for transaction routing is identical to that for CICS function request shipping. (See the sections "System Entries in Terminal Control Table" on page 249 and "Terminal Control Support for CICS Function Request Shipping" on page 265.)

In order to support transaction routing, the relay transaction owns two TCTTEs; see Figure 88 on page 270. One TCTTE is that for the terminal, the other TCTTE is that for the link to the user transaction. The link TCTTE has bit TCTERLT in field TCTETSU set on to indicate that it is being used by the relay transaction.

The user transaction owns two or more TCTTEs; see Figure 89 on page 270. One TCTTE is always present for the link to the relay transaction, and another TCTTE, called the surrogate TCTTE, represents the terminal TCTTE in the relay transaction address space. The field TCTTERLA in the surrogate TCTTE contains the address of the TCTTE for the link to the relay transaction. Bit TCTESUR (in the field TCTETSU) set on indicates that the TCTTE is for a surrogate terminal. The link TCTTE has bit TCTERLX in field TCTETSU set on to indicate that it is being used as a relay link.

If the user transaction executes CICS functions which are shipped to another address space or processing system, then a TCTTE is created for each different address space or processing system.

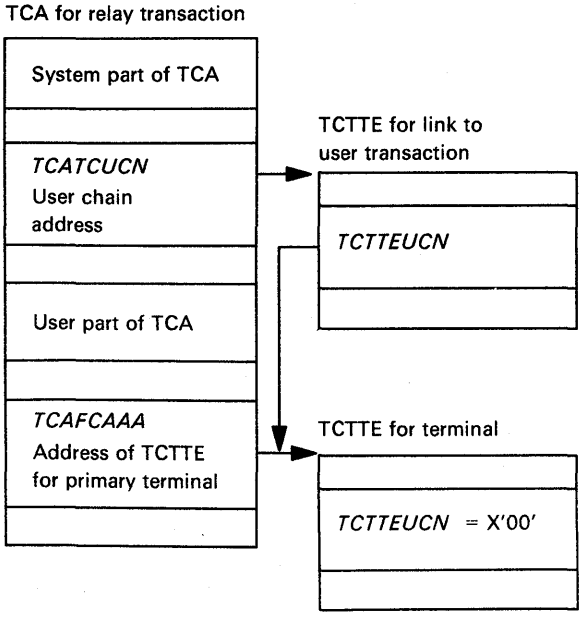


Figure 88. Control Blocks for Relay Transaction

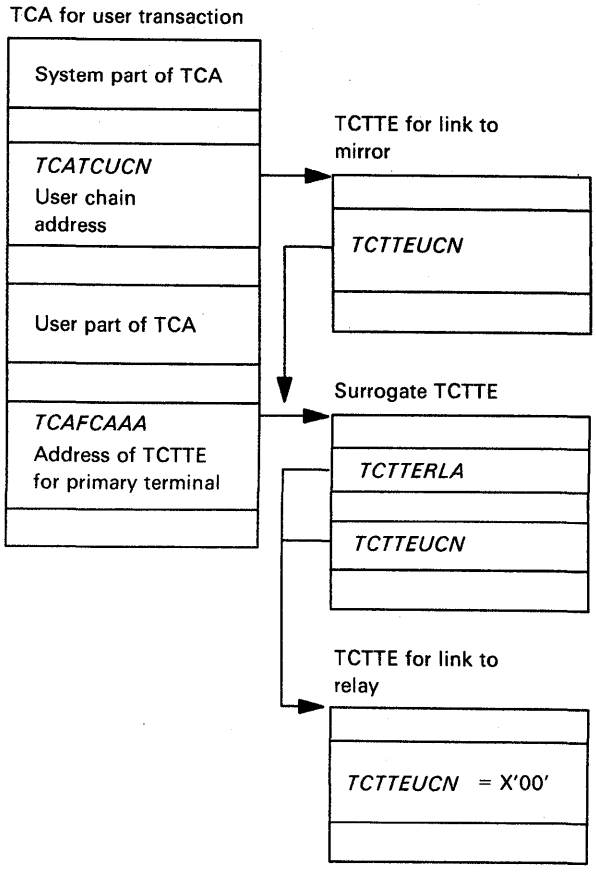


Figure 89. Control Blocks for User Transaction

Execution of Transaction Routing

When an operator enters a transaction code for a transaction defined as remote in the PCT, then that transaction is attached to the terminal and the CICS relay program, DFHCRP, is invoked. DFHCRP extracts from the PCT the identification of the remote system on which the transaction resides, and the name of the transaction in that system. DFHCRP then issues a DFHIS TYPE=ROUTE macro. This macro invokes DFHZTSP (the terminal sharing program, in module DFHZCX) to route the transaction to the required system.

If you are using ISC for transaction routing, DFHZTSP uses the profile specified in the PCT. For ISC and MRO, DFHZTSP allocates a link TCTTE for the remote system and invokes DFHXTP (the transaction routing data transformation program) to generate the data that is to be shipped. DFHXTP gets storage for a TIOA chained from the link TCTTE. An attach FMH is built in this TIOA and a transaction routing FMH is concatenated with it. The transaction routing FMH is followed by the transaction routing data which consists of selected fields from the operator's terminal TCTTE and the input message. Having built the data stream, DFHXTP returns control to DFHZTSP.

DFHZTSP issues a DFHTC TYPE=CONVERSE macro against the link TCTTE, and waits for a reply from the remote system. The data is transferred in the same way as for CICS function request shipping, as described in the section "Communication with a Remote System" on page 249.

In the remote system, the user transaction code is extracted from the attach FMH by DFHZATT (task attach, in DFHZCP) and the user task is attached with the link TCTTE as its principal facility. Control is passed to DFHZSUP (startup program, in DFHZCP) which issues a DFHIS TYPE=RATT macro. This macro invokes DFHZTSP to build a surrogate TCTTE.

It does this by allocating shared storage for the TCTTE (and TCTUA) and copying into it a model TCTTE that contains the static description of the terminal as specified by the definition of the remotely owned terminal in the TCT. DFHZTSP

invokes DFHXTP to merge the dynamically changing data that was passed with the attach request into the surrogate TCTTE, and to build a TIOA containing the input message. DFHZTSP then sets up security fields and priority values and connects the now complete surrogate TCTTE to the user task as its principal facility.

Control is returned to DFHZSUP, which performs security checks, and transfers control to the user application program. Thus, in the remote system, the user application executes with a surrogate TCTTE that provides the same interface as the real TCTTE to the application program.

When the application program issues a terminal control request (either directly, or as a result of a BMS or batch data interchange request) control passes to DFHZARQ (application request, in DFHZCP). If the request is issued against a surrogate TCTTE, DFHZARQ issues a DFHIS TYPE=RARQ macro that invokes DFHZTSP to ship the request to the relay transaction.

It does this by first invoking DFHXTP to package the terminal control request, relevant TCTTE information, and the output TIOA, if any, into a transaction routing FMH in a TIOA chained from the link TCTTE. DFHZTSP then issues a DFHTC TYPE=CONVERSE against the link in reply to the CONVERSE issued by the relay transaction.

In the local system, the relay transaction receives the reply to the CONVERSE issued on its behalf by DFHZTSP. DFHZTSP invokes DFHXTP to unpack the request. DFHZTSP then determines if there is a terminal control request to be issued against the operator's terminal, and, if so, issues the required the request thereby executing the same terminal control request as was issued by the user transaction. When control is returned to DFHZTSP after the terminal control request has been executed, DFHZTSP invokes DFHXTP to package the result of issuing the request into a TIOA on the link. DFHZTSP then sends this data across the link by issuing a DFHTC TYPE=CONVERSE in reply to the CONVERSE issued on behalf of the user transaction.

This process is repeated until the user transaction is terminated. When a transaction is terminated, its principal facility is released by a DFHTC TYPE=FREE macro issued either by DFHKCP or by DFHSPP.

This macro invokes DFHZISP (intersystem program, in DFHZCP) to free the specified TCTTE. If the TCTTE to be freed is a surrogate TCTTE, DFHZISP issues a DFHIS TYPE=RDET macro to invoke DFHZTSP to detach the surrogate. DFHZTSP once again invokes DFHXTP to package any outstanding terminal control request and other relevant information (including the TCTUA value) into a link TIOA. DFHZTSP then issues a DFHTC TYPE=(WRITE, LAST) macro to ship the information to the relay transaction and terminate the conversation

When this last flow is received by the relay transaction, DFHZTSP unpacks the data and issues the terminal control request (if any) as before, but this time, because the end of conversation indicator has been received, DFHZTSP returns control to DFHCRP. DFHCRP issues a DFHPC TYPE=RETURN macro to terminate the relay transaction.

Figure 90 shows the flows between the terminal and the user transaction when the terminal enters one message, and the user transaction issues one message to the terminal and waits for its completion.

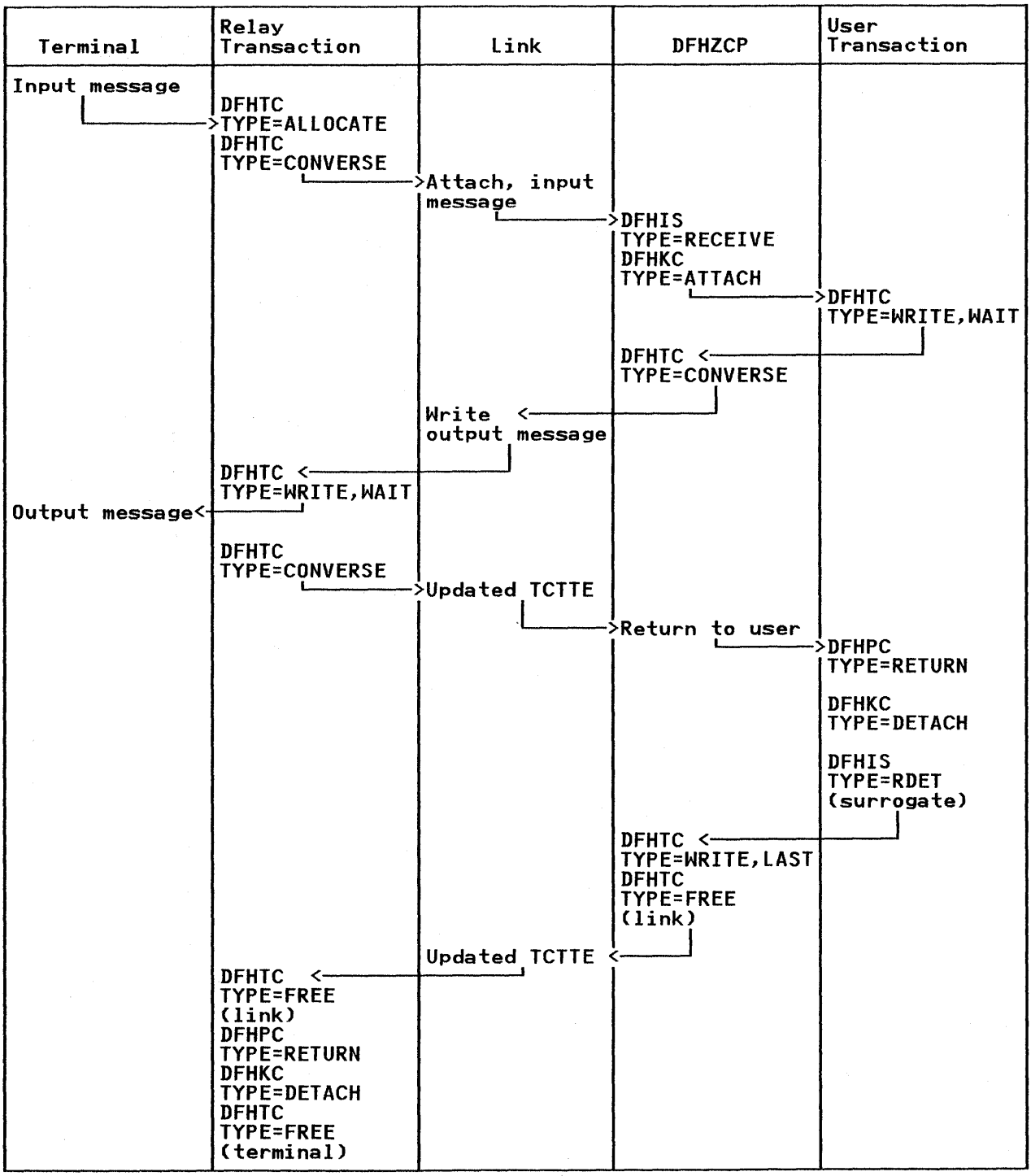


Figure 90. Flow for Message, Reply, and Wait

Figure 91 shows the flows in the same situation when the user transaction does not wait for completion of the request. The transmission from the user transaction does not take place until a WAIT or SYNCPOINT is issued or the user transaction terminates.

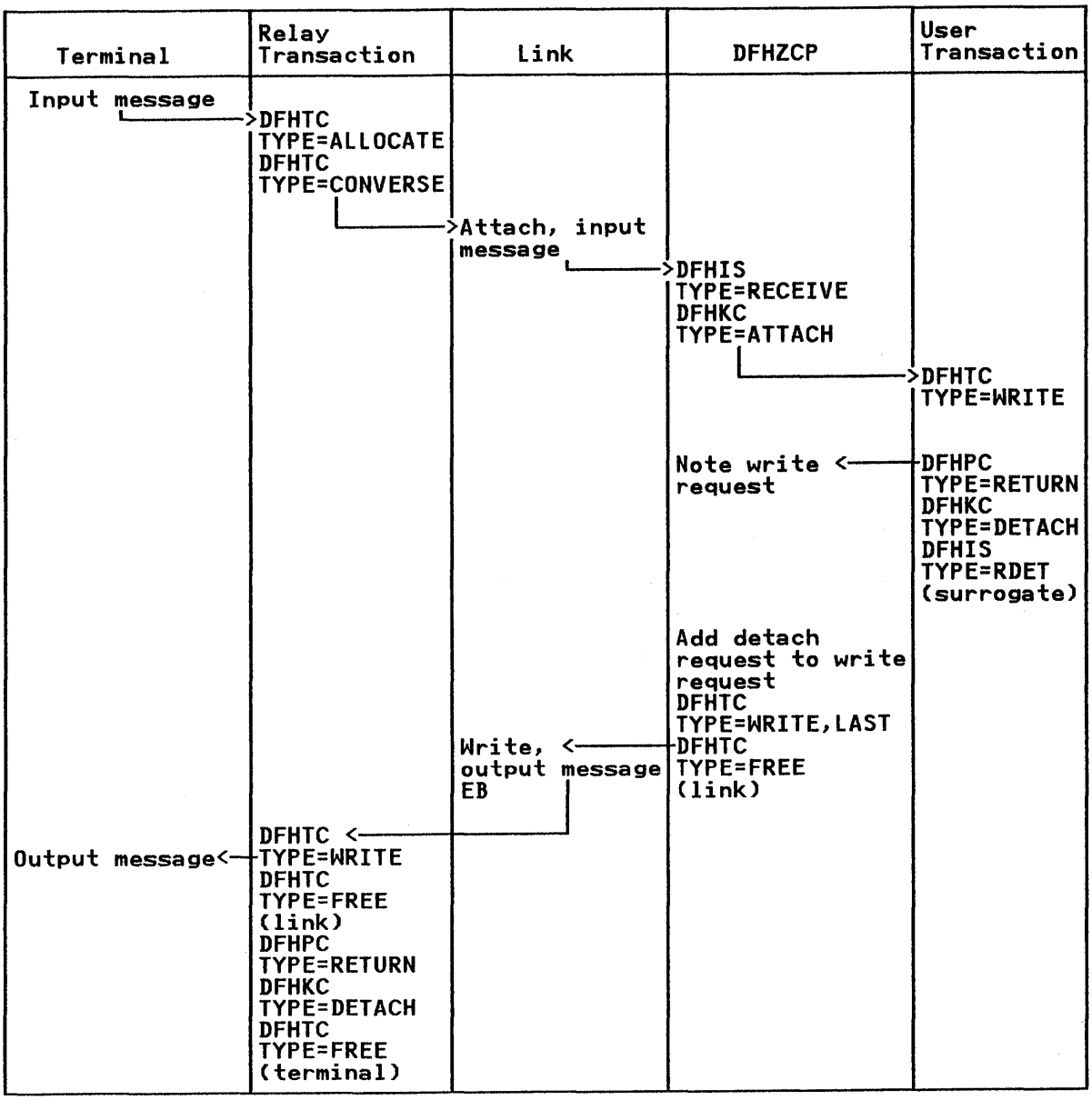


Figure 91. Flow for Message and Reply (No Wait)

Daisy Chaining

When a terminal request is routed to a remote region, it is possible that the transaction is defined to be remote to this region also. In this situation, known as **daisy chaining**, a chain of relay transactions is created.

Recovery and Restart

Abnormal Termination

Abnormal termination can be caused by:

- The user transaction abending
- The relay transaction abending
- The session going down
- The transaction being disabled.

If the user transaction abends, then DFHACP sends a message to CSMT, but does not send a message to the terminal. DFHACP does not write to surrogate TCTTEs. DFHSPP sends ABORT to the relay transaction and the relay transaction is abended. DFHACP on the terminal owning system sends a message to CSMT, and also to the terminal. If using ISC, an ATNI abend code is sent to the application programmer.

If the relay transaction abends, then DFHACP sends a message to CSMT and to the terminal. DFHSPP sends ABORT to the user transaction and the user transaction is abended. DFHACP on the transaction owning system sends a message to CSMT.

If the session goes down, then the transaction waiting on the link is abended immediately. DFHACP sends a message to CSMT, and, if running on the terminal owning system, to the terminal. DFHSPP does not send ABORT. The other transaction has control, and the next time it tries to use the link, it is abended. DFHACP sends another message to CSMT, and, if running on the terminal owning system, to the terminal.

If the transaction is disabled, or CSAC is run instead of the user transaction for any reason, DFHACP sends ABORT to the relay transaction which abends causing DFHACP to issue messages to CSMT and to the terminal.

Recovery

Recovery for terminal control operations is only required if the transaction specifies that message protection is required. All logging and recovery processing for committed output messages is done by the terminal owning system. When the user transaction takes a sync point (either a user SYNCPOINT or a RETURN), then the requirement for a sync point is shipped to the relay transaction (provided the user task has been defined as requiring message protection). The relay transaction then takes its own sync point.

If using ISC, deferred flows on the relay link associated with the surrogate TCTTE are sent, using the DFHIS TYPE=RFLUSH macro.

The method of synchronizing the commitment of changes to local and remote resources and committed output messages is the same as that used for CICS function request shipping (see "CICS Function Request Shipping Handling of Sync Point Requests" on page 259).

Transaction Restart

Transaction restart for user transactions connected to remote terminals is supported provided there has been no terminal traffic other than the initial input message.

Transaction restart involves a sync point following the abend and backout. This sync point causes ABORT commands to be issued to all related remote transactions causing them to backout. However, ABORT is not sent to the relay transaction because that would cause both the terminal and the link to be freed.

Figure 92. Format of CICS Transaction Routing Data

Security

The user can define a security key for the relay transaction. This is the TRANSEC value, which is specified in the PCT entry defining the transaction as remote. Normal security checking is performed by DFHXSP when the relay transaction is attached.

The user can also define a security key for the SYSTEM entry of the TCT, which is used to describe the properties of the link between connected systems. When the user transaction is attached, DFHISP performs the usual security checks against the security key, OPERSEC, of the link. OPERSEC must therefore match the user transaction security key, otherwise that transaction cannot be invoked by the link.

Logical Unit Type 6.1 Protocol

The logical unit type 6.1 form of the data stream is used for ATTACH, SPR, PREPARE, COMMIT and ABORT. The type 5 attach header specifies the user transaction name as the process and CSRR as the return process.

For other requests, the CICS private FMH (type X'43') is used. The groups and function bytes of the FMH are used for the primary function information. The TCA request bytes and the TCTTE and TIOA information are passed as a series of variable-length parameters in the form shown in Figure 92.

Each data item is preceded by a type byte. The type byte is followed by a 1-byte or 2-byte length field, depending on the type byte.

Distributed Transaction Processing for ISC and MRO

Distributed transaction processing is the ability of a CICS system to initiate and communicate with a transaction in another system which has a compatible interface. This section describes what happens when the interface with the other system conforms to the logical unit type 6.1 (LU6.1) protocol, or the other system is CICS in a different address space using MRO.

The data and control information for the conversation between systems uses an SNA (VTAM) LU6.1 session. The session is represented within CICS by a TCTTE with an LU6.1 terminal type. The basic operations are initiating, terminating, and continuing a conversation. To perform these operations, CICS uses terminal control commands, and these commands are mapped onto SNA control indicators and data flow commands.

Messages are transmitted between applications in units of data called request/response units (RUs). The SNA control indicators are transmitted in a header that precedes each RU and is called the request/response header (RH). The control indicators used in this section are as follows:

```
BB   -  begin-bracket
EB   -  end-bracket
RQD1 -  request-definite
RQE1 -  request-exception
RQD2 -  sync-point-request
CD   -  change-direction.
```

The meaning and use of these indicators are explained later as they are referred to. The CICS application program controls these indicators by specifying options in CICS commands.

The conversation corresponds to one SNA bracket; BB is the first indicator transmitted and EB is the last indicator transmitted defining the beginning and end of a conversation respectively.

Contention between transactions at each end of the session competing for use of the session is resolved by using half-duplex flip-flop protocol, which allows transmission only one way at a time once a conversation has been started.

Logical units of work in the two systems are coordinated by using RQD2 indicators.

Negotiable Bind for Distributed Transaction Processing

Negotiable bind for distributed transaction processing is the same as that for function requesting shipping, see "Negotiable Bind for Function Request Shipping (VTAM Only)" on page 248 for details.

Normal Flows

The transaction which starts a conversation is called the front-end transaction; the transaction created by the conversation is called the back-end transaction. After the initial message has been sent and the back-end transaction created, both transactions have equal status and are able to send or receive messages.

In order to control which transaction is sending and which is receiving at any given time, the half-duplex flip-flop protocol is used. This means that transmission is allowed in only one direction at a time. The front-end transaction starts in the send state and the back-end transaction starts in the receive state. To change the status, a CD indicator must be sent by the transaction which is in send state. In this way, the transactions at each end of a conversation keep in step with each other.

At the conclusion of a RECEIVE command, the state of an LU6.1 session is contained in the EXEC interface block (EIB). The field EIBRECV is X'00' for **not-in-the-receive-state** and X'FF' for **in-the-receive-state**. Any attempt to send by a transaction which is in the receive state is an error.

The terminal control commands cause certain SNA indicators to be sent. The first SEND command generates a BB indicator which starts a conversation. The LAST option in the SEND command generates an EB indicator to terminate a conversation.

When the SEND command is used with the INVITE option, a CD indicator is sent with the message to allow the other system to send a reply. When the SEND command has a DEFRESP option, an RQD1 indicator is sent. The RQD1

indicator means that the other LU6.1 system must send a response that it has processed the message.

The SYNCPOINT command generates a stream containing an LUSTAT(0006) command with RQD2 indicator (provided only one session is active), otherwise an FMH PREPARE is sent. The RECEIVE command, if issued while a system is in the send state, generates an LUSTAT(0006), an RQE1 indicator, and a CD indicator. The RQE1 indicator means that a response is only expected if an error has occurred. The RETURN command generates an LUSTAT(0006), an RQD2 indicator, and an EB indicator.

For multiregion operation, you cannot execute the sequence **SEND, RECEIVE**, but must use the INVITE option with the SEND statement. Also, you cannot execute the sequence **SEND INVITE, SYNCPOINT, RECEIVE**.

Logical Unit of Work

The CICS transactions involved in an LU6.1 conversation are not only sending data, but also making alterations to permanent storage (for example, files, data bases). If these resources are recoverable, then the changes made at each end of the session must be coordinated; either both sets of changes occur or none occur.

This means that the sync points and abends which would occur independently in each system must be communicated to the other system. This is achieved by using the RQD2 indicator, or the FMH PREPARE. The logical unit status PREPARE, negative responses, and error processing messages indicate a failure.

The CICS transaction knows about a sync point through the SYNCPOINT command in the sending system or the EIBSYNC byte in the EIB in the receiving system (EIBSYNC has the value X'FF' for a **sync-point-required** condition, X'00' otherwise). A transaction may initiate a SYNCPOINT only when in send mode. A SYNCPOINT command causes EIBSYNC to be set to X'EF' in the other transaction, and this other transaction must issue a SYNCPOINT command before it can send any data. This action sets EIBSYNC equal to X'00', and the conversation can continue. Because transaction termination implies a sync point, this is a valid response to the EIBSYNC condition, provided that either EIBFREE is indicated or the transaction is in send mode. EIBFREE is a byte in the EIB which has the value X'FF' indicating that the FREE command should be issued, or X'00' otherwise.

If a transaction detects an error, then by issuing an ABEND command, the fact of the error occurring is transmitted to the other transaction involved in the conversation. The latter transaction receives the error as an ATNI abend code. The recovery required to clear the error from the session is done by CICS.

ISSUE SIGNAL Command (ISC only)

In a conversation between two transactions, only one transaction can be in the send state at any particular time; the other transaction must be in the receive state, and must continue to issue RECEIVE requests until the other transaction decides to change the status.

If the receiving transaction wishes to send data rather than receive data, then it can inform the sending transaction by executing the ISSUE SIGNAL command. The sending transaction is informed of the signal when it next issues a SEND or RECEIVE command provided a HANDLE CONDITION SIGNAL has been executed previously. Alternatively, the sending transaction can look at the field EIBSIG in the EXEC interface block (EIB). This field is set to X'FF' when a signal is received. The sending transaction must then decide whether or not to honor the signal request.

Sessions

Starting a Conversation

The front-end transaction has three things to do in order to begin a conversation:

1. Acquire an LU6.1 session.
2. Cause the creation of the transaction required as the back-end to this conversation.
3. Optionally, send the back-end transaction its first input message.

Steps 2 and 3 are usually combined into a single SEND or CONVERSE command. The back-end transaction is defined either by the use of a field in the ATTACH function management header (FMH) or by the first four bytes of user data. The FMH is a variable-length field that can be sent between logical units via the TIOA. When present, the FMH is at the front of the TIOA.

Terminating a Conversation

To terminate a conversation, the transaction does two things:

1. Sends the EB indicator to the other transaction.

2. Releases the LU6.1 TCTTE for use by other transactions.

The EB indicator is sent by using the LAST option in the SEND command, or issuing the FREE command while in send state. The FREE command releases the LU6.1 TCTTE. Transaction termination always terminates a conversation by an implied FREE command.

An example of the flows for a simple conversation is given in Figure 93 on page 279.

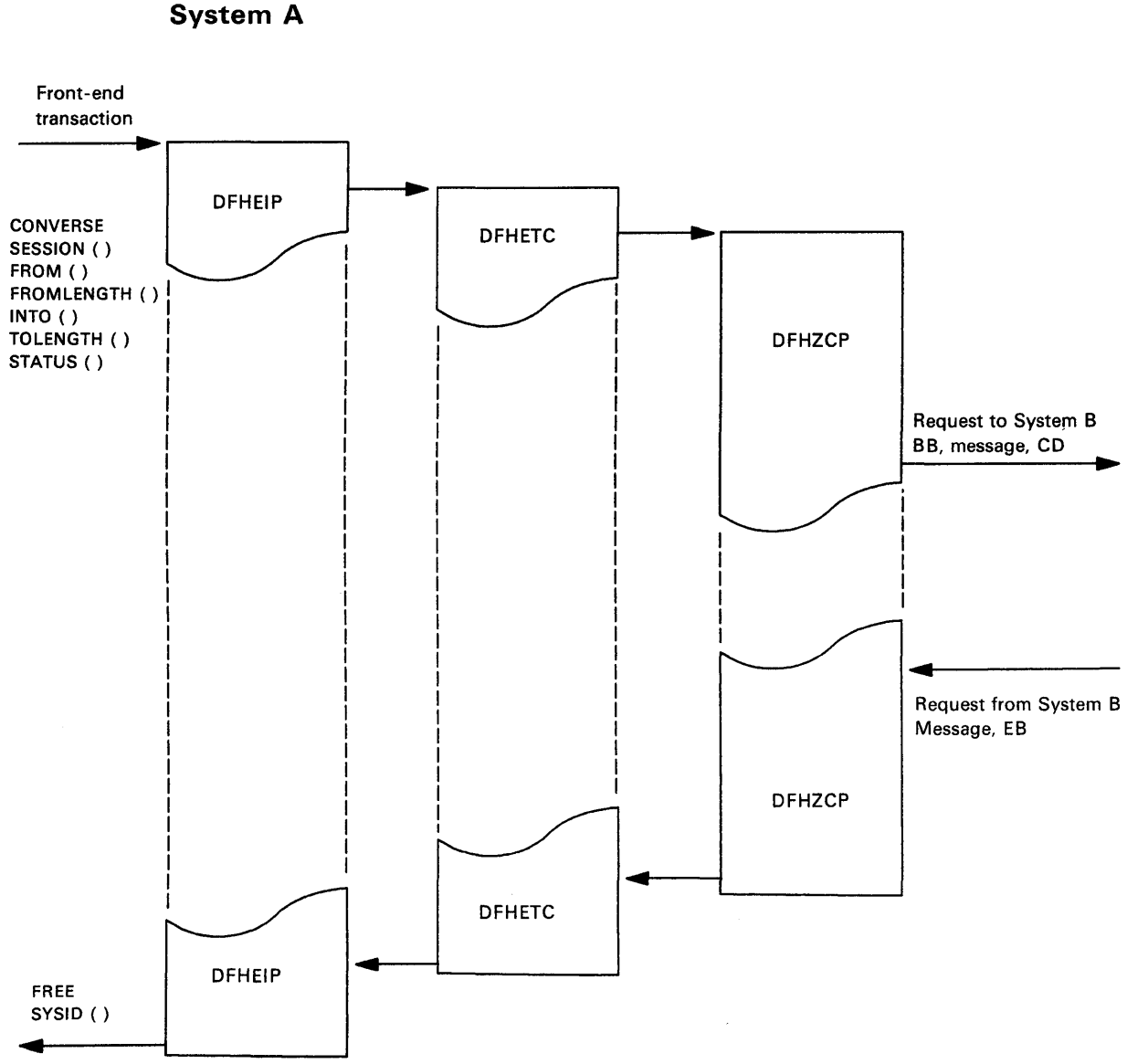


Figure 93 (Part 1 of 2). Distributed Transaction Processing – Simple Conversation

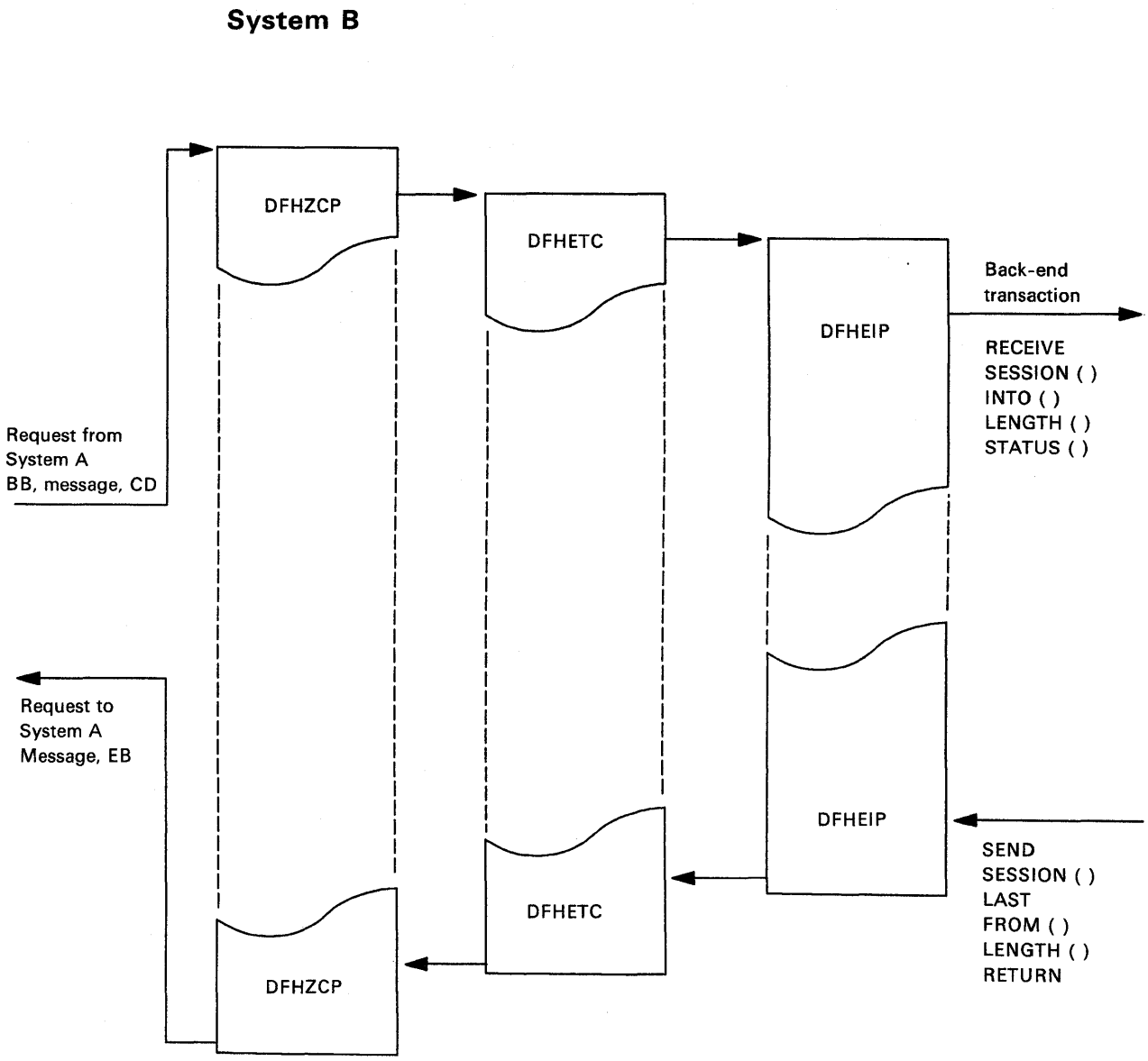


Figure 93 (Part 2 of 2). Distributed Transaction Processing – Simple Conversation

Distributed Transaction Processing Sync Points and Failures

CICS handles session failures and systems failures for distributed transaction processing in the same way as for CICS function request shipping. See the sections "CICS Function Request Shipping Sync Points and Session Failures" on page 262 and "CICS Function Request Shipping Sync Points and System Failures" on page 264.

Interregion Component

Interregion communication (IRC) enables communication between CICS systems in the same processing system (that is, multiregion operation), or between CICS systems and shared data base batch regions.

The modules for IRC are:

1. The interregion programs, DFHIRP, and DFHXMP. DFHIRP performs initialization

and termination for DFHXMP. When MVS cross-memory services are requested (ACCMETH=(IRC,XM) in the TCT), DFHXMP is used; otherwise, communication is performed by DFHIRP running as an SVC.

2. CICS address space modules – DFHCRC, DFHCRNP, DFHCRR, DFHCRSP, DFHZCP, and DFHZCX. These modules provide the CICS address space with a DFHTC-level interface to interregion communication (in the same way as DFHZCP provides a DFHTC-level interface to VTAM). This enables other CICS modules (such as DFHISP) to allocate and execute input/output operations on IRC sessions. The IRC sessions are used for all forms of IRC communication, and the macro-level services available for IRC are broadly the same. Thus DFHISP works both for IRC and ISC CICS function request shipping.

MVS Cross-Memory Program (DFHXMP)

When the MVS cross-memory services are used for interregion communication, the **switch** and **pull** functions are performed by DFHXMP which is entered by issuing a **program call** (PC) instruction instead of an SVC. DFHXMP does not need a commonly addressable buffer or service request blocks (SRBs) to effect data transfer between address spaces.

Code in DFHIRP performs the cross-memory initialization and termination functions for DFHXMP as follows:

LOGON: Acquire and initialize the cross-memory resources (authorization index (AX), linkage index (LX), and entry table (ET)), unless this has already been done by a previous logon in this address space.

CONNECT: Update the authority tables (ATs) of both address spaces to allow each one to establish addressability to the other, unless this was done when a previous connection was established between them.

DISCONNECT: If the last cross-memory connection between a pair of address spaces is being removed, update the caller's AT so that the other system is no longer permitted to access the caller's address space.

LOGOFF: Free the cross-memory resources acquired by logon if they are no longer required by the caller's address space.

CICS Address Space Modules

The CICS address space modules control the handling of requests to and from other address spaces. The functions of each module are as follows:

DFHCRSP – CICS IRC Startup Module
Execution of this module makes interregion communication possible between this address space and other address spaces. DFHCRSP, which can be invoked either at system initialization or by the master terminal, issues a LOGON request to the SVC routine and attaches the transaction CSNC (connection manager program DFHCRNP).

DFHCRNP – Connection Manager (Transaction CSNC)
The main purpose of CSNC is to perform housekeeping and control on IRC sessions. Its functions include the following:

1. Establish connections to other address spaces (by issuing CONNECT requests).
2. Detect unsolicited input data on connections and attach requested tasks to process such data.
3. Disconnect unallocated (**between-bracket**) sessions during QUIESCE.
4. Issue DFHKC AVAIL for any secondary sessions which have become available for reallocation, and are in demand.
5. Issue LOGOFF when QUIESCE is complete.

CSNC is attached by DFHCRSP (IRC startup), and then suspends itself. It is resumed by DFHKCP if DFHKCP decides that CSNC has work to do.

DFHCRR – CICS Session Recovery Module (Transaction CSIR)
Whenever a new connection is established (via a successful CONNECT request), the first transaction to be run on the connection is CSIR. DFHCRNP attaches CSIR at the secondary end of the connection (that is, at the source of the connection). CSIR sends a data stream down to the other end of the connection (the primary end) that causes CSNC to attach CSIR at the primary end. The purpose of these two CSIRs is to exchange information in order to determine whether either end of the connection was **in-doubt** when the previous use of the connection was terminated, and, if so, whether the two ends were in-sync or out-of-sync (see "CICS Function Request Shipping Sync Points and Session Failures" on page 262).

DFHZCX (DFHZIS2) – CICS Terminal Control Module
DFHZCX contains several routines:

I/O Request Routine (IORENT)
Provides a WRITE/WAIT/READ interface to interregion connections.

GETDATA Routine (GDAENT)
Retrieves input data from an IRC connection and puts it into a TIOA.

RECEIVE Routine (RECENT)
Receives unsolicited data (**begin-bracket** in SNA terms) and checks validity.

DISCONNECT Routine (DSCENT)
Cleans up this end of a connection, and issues DISCONNECT request to DFHIRP.

OPRENT Routine (OPRENT)
Issues an INSRV request to DFHIRP, in order to allow future connections between this subsystem and a specified subsystem.

RECABRT Routine (RCAENT)
Is invoked when an ABORT FMH (FMH07) is received (indicating that the connected transaction has abended). The routine issues a message describing the failure.

STOP Routine (STPENT)
Is invoked when communication with other address spaces is to be terminated. The routine issues a QUIESCE request to DFHIRP.

LOGOFF Routine (LGFENT)
Is invoked when quiesce is complete (and during system termination and abend processing). The routine issues a LOGOFF request to the SVC routine.

DFHZCX also contains routines representing terminal control services which are supported by IRC (in common with VTAM). These routines include ALLOCATE, FREE, PREPARE, SPR, COMMIT, ABORT.

DFHZCP – CICS Terminal Control Module
Copybook DFHZARQ is used (in common with all other telecommunications access methods) to handle WRITE/WAIT/READ-level requests against IRC connections (sessions). Routine TCZAQI00 in DFHZARQ specifically handles IRC requests by performing SNA request header processing and invoking IORENT (see DFHZCX) in order to perform the I/O on the session.

DFHCRC – Interregion Abnormal Exit Module
DFHCRC contains an ESTAE exit to terminate IRC in abnormal conditions; this exit is established **before** the ESTAE exit in DFHSRP (system recovery program). DFHCRC invokes the IRC CLEAR service.

Shared Data Bases Between CICS and IMS/VS Batch Programs

So that an IMS/VS DL/I data base can be accessed by CICS **and** by concurrently active IMS/VS batch job steps in other address spaces, all requests for access to the data base are coordinated in one address space – the CICS address space.

A DL/I request from an IMS/VS batch application program is handled, in the first instance, by a CICS-DL/I batch region controller (one in each batch region) instead of by the IMS/VS DB batch region controller. The CICS-DL/I batch region controller passes the DL/I request across to the CICS address space by means of the IRC SVC routine. In the CICS address space, a mirror transaction handles the request as if the request had come from an IRC-connected CICS system. The DL/I response is subsequently returned to the calling batch region, again by means of the IRC routine.

Components of the Shared Data Base Mechanism

Shared data base is implemented by using the IRC mechanism, in conjunction with the CICS-DL/I Batch Region Controller Modules (DFHDRP, DFHDRPA-G).

These modules establish the batch region environment and control the batch region functions that enable an IMS/VS application program to access the DL/I data base via CICS.

CICS-DL/I Batch Region Controller Modules (DFHDRP, DFHDRPA-G)

The DFHDRP modules control the execution of the shared data base job step in the batch region (see Figure 94 on page 291). The functions of each module are as follows:

DFHDRP – Batch Region Control Module
DFHDRP is the first module to be executed during a batch shared data base job step. The module resides in the link pack area and is APF (authorized program facility) authorized. DFHDRP opens the authorized program library DFHLIB which contains the remainder of the batch region controller modules; it then passes control to DFHDRPA.

DFHDRPA – Batch Region Initialization Module
Processes the user parameters in the JCL EXEC statement and establishes a connection with the CICS address space (via DFHDRPF); it then passes control to DFHDRPB.

DFHDRPB – Application Program Control Module
Initiates execution of the batch application program. When the application finishes, DFHDRPB invokes DFHDRPC.

DFHDRPC – Batch Region Termination Module
Invokes the CONVERSE routine DFHDRPE to send a sync point request (SPR) to CICS to indicate that DL/I activity is complete; DFHDRPC then invokes the cleanup routine in DFHDRPD to disconnect from the CICS address space, and to logoff. DFHDRPC finally returns control to OS/VS at job step completion.

DFHDRPD – Batch Region Cleanup and Exits Module
Consists of three routines – a cleanup routine, the ESTAE exit routine, and the SPIE exit routine. The cleanup routine (which is invoked when the batch job step is to terminate either normally or abnormally) issues a DISCONNECT request and a LOGOFF request to the SVC routine. The ESTAE and SPIE exits invoke the cleanup

routine before continuing abnormal termination.

DFHDRPE – DL/I Request Handling Module
DFHDRPE has two routines:

Program Request Handler Routine
Receives control, via the language interface routine (DFSLI000), when the batch application program issues a DL/I request; for data base **access** requests, it then invokes the CONVERSE routine in order to send the request to CICS and await a reply.

CONVERSE Routine
Invokes DFHXFQ (which is link-edited with DFHDRPE) to transform the DL/I argument list into a format which can be processed by the CICS system. The formatted data is sent by issuing a SWITCH request to the SVC routine. When the SVC posts the batch SLCB ECB to indicate that the reply has arrived, the CONVERSE routine again invokes DFHXFQ to transform the reply into the format required by the user.

DFHDRPF – SVC Initialization Module
Is invoked by DFHDRPA (and by DFHDRPE for CHECKPOINT calls). DFHDRPF issues a LOGON request and a CONNECT request to the SVC routine to establish a connection with CICS; it then sends (via the CONVERSE routine in DFHDRPE) a DL/I scheduling request for the user's PSB, so that CICS can schedule the PSB.

DFHDRPG – EXEC Interface Stub
Receives parameters of the command and builds a list to pass to DFHDRPE. On return from DFHDRPE, the status code in the PCB is examined and action taken for errors.

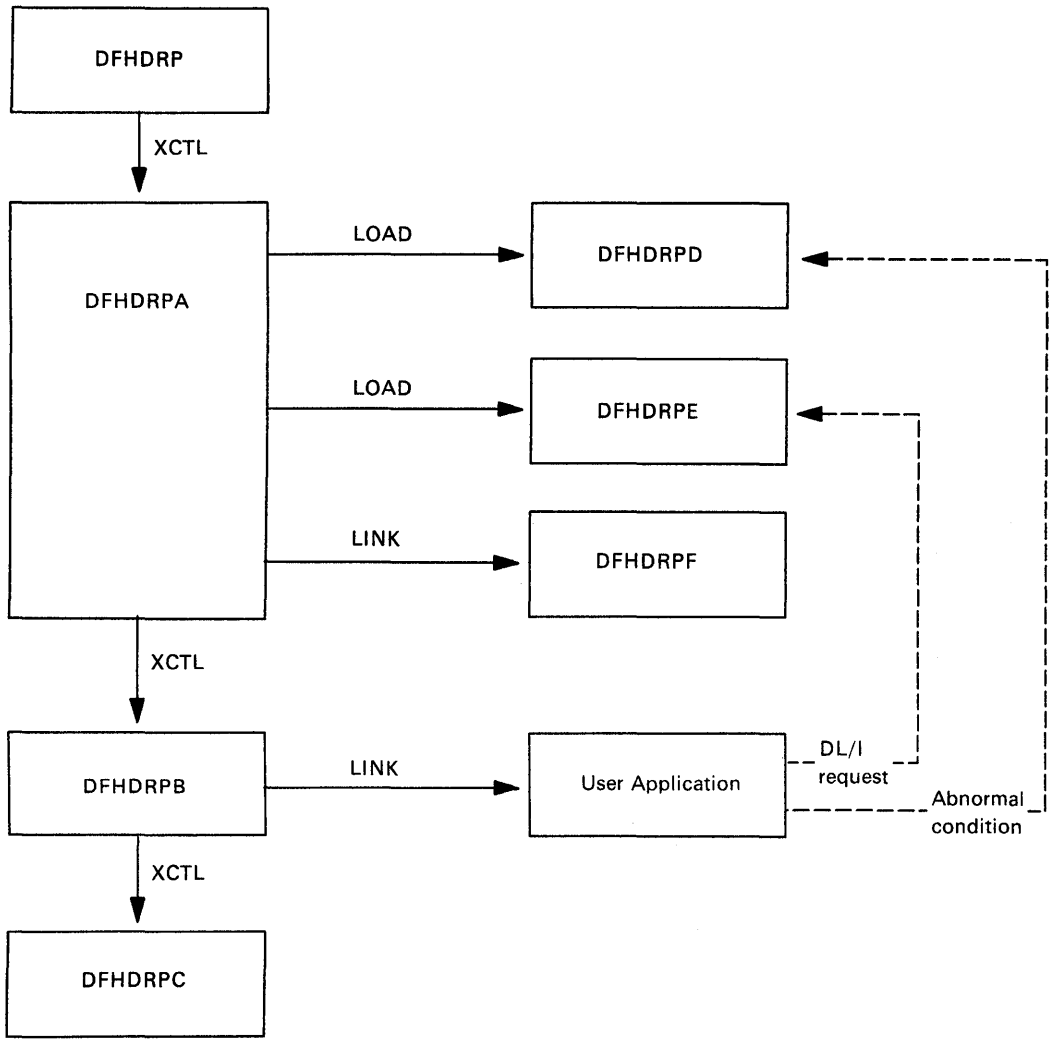


Figure 94. Interregion Communication – Batch Region Modules

Handling of DL/I Requests from a Batch Region

The following paragraphs describe the sequence of actions involved in the handling of DL/I requests from an IMS/VS application.

A. CICS Address Space Makes Interregion Communication Available

Before a batch region can pass DL/I requests to CICS, CICS must make the interregion communication (IRC) facility available. It does this at system initialization time, or by a CEMT or CSMT command, as follows:

1. DFHCRSP is invoked and issues a LOGON request to the SVC routine, which creates the CICS address space's control blocks (including an SLCB, and the required number of SCCBs).
2. DFHCRSP attaches transaction CSNC (connection manager), which suspends itself until a batch region issues a CONNECT request.

B. Batch Region Job Step Starts

A job step that requires access to IMS/VS DL/I has an EXEC statement that causes the program DFHDRP to be executed:

1. DFHDRP transfers control to DFHDRPA, which loads DFHDRPD and DFHDRPE, and links to DFHDRPF.

2. DFHDRPF establishes a connection with the CICS address space by issuing a LOGON request and a CONNECT request to the SVC routine. (The LOGON request causes the SVC routine to create an SLCB and one SCCB to represent the connection in the **batch** region; the CONNECT request allocates the batch region's SCCB and one of the CICS address space's SCCBs.)

3. DFHDRPF initiates the DL/I activity by sending to CICS (via DFHDRPE) a DL/I argument list that represents a scheduling request for the user's PCB.

4. The transmission is effected by the CONVERSE routine in DFHDRPE which:

   a. Transforms the DL/I request into a format which can be processed in the CICS address space; this transformation is similar to the **transformation 1** for CICS function request shipping referred to in Figure 83 on page 257.

   b. Transmits the transformed request by issuing a SWITCH request to the SVC routine.

5. DFHDRPE waits for the response.

C. CICS Address Space Handles DL/I Scheduling Request

When a batch region issues a CONNECT request, the SVC routine (1) OS-posts the SLCB's ECB to indicate that there is input from another address space to be processed, and (2) sets the post bit in the ECB of the newly allocated SCCB to indicate that there is activity on this **particular** connection.

Because the SLCB's ECB is now posted, DFHKCP finds (from a status flag in the SLCB) that there is a new connection and resumes the CSNC (connection manager) transaction.

The CSNC transaction attaches the mirror transaction (CSMI) which then handles the DL/I request on behalf of the batch application program as if the request had come from an ISC-connected system.

That is:

1. The mirror transaction invokes DFHXFX to transform the received request into a form suitable for execution by DL/I (transformation 2).

2. The mirror transaction passes the request to DL/I (module DFHDLI).

3. When DL/I has successfully completed the request, control is returned to the mirror transaction with the results in the user interface block (UIB).

4. The mirror transaction then invokes DFHXFQ to transform the results into a form suitable for transmission to the batch region (transformation 3).

5. The mirror transaction issues a terminal control request (WRITE,WAIT,READ) to transmit the reply and to await the next request.

6. In terminal control, DFHZCP transmits the reply to the batch region by invoking the I/O routine in DFHZCX, which makes a SWITCH request to the SVC routine. (DFHZCX then waits on the connection's SCCB ECB for the next request from the batch region.)

D. Batch Region's Application Program Is Invoked

On receipt of the reply from the CICS address space:

1. The CONVERSE routine in DFHDRPE invokes DFHXFQ to transform the reply into the format required by the batch application program; this transformation is similar to the **transformation 4** for CICS function request shipping referred to in Figure 83 on page 257.

2. Control then returns (via DFHDRPF) to DFHDRPA.

3. DFHDRPA transfers control to DFHDRPB which returns to the application program.

The application program then executes. Each DL/I request is intercepted by IMS/VS's Language Interface Routine (link-edited with the application program), which invokes the Program Request Handler routine in DFHDRPE.

DFHDRPE transforms the request and transmits it to the CICS address space by means of a SWITCH request (as described for the scheduling request in steps B.4 and B.5).

E. CICS Address Space Handles DL/I Requests

The mirror transaction, which is still an active task serving the connection, is now dispatchable because the SCCB ECB on which it is waiting (see step C.6) is now posted; it handles each DL/I request as described for the scheduling request (step C.1 onwards).

F. Batch Region Terminates

When the application program terminates, DFHDRPB transfers control to DFHDRPC. DFHDRPC sends to CICS a sync point request (SPR) via the CONVERSE routine in DFHDRPE to indicate that DL/I activity is complete. On receipt of a positive response, DFHDRPC finally invokes the cleanup routine in DFHDRPD to issue DISCONNECT and LOGOFF requests to the SVC routine, and the batch job terminates.

G. CICS Address Space Disconnects

On receipt of the batch region's SPR, the mirror task that has been serving a particular connection terminates and control is passed to the sync point program DFHSPP. DFHSPP sends the positive response back to the batch program and then invokes the FREE routine in DFHZCX which issues its own DISCONNECT request to the SVC routine.

H. CICS Makes Interregion Communication Unavailable

Steps B through G describe the events that occur during the lifetime of a single batch region. In practice, these events may occur concurrently for several batch regions.

Interregion communication is closed down (for example by the CEMT or CSMT transaction) by invoking the STOP routine in DFHZCX. The STOP routine issues a QUIESCE request to the SVC routine which posts the ECB in the CICS address space's SLCB when all connections are broken. Because the SLCB's ECB is now posted, DFHKCP finds (from a status flag in the SLCB) that quiesce is complete and resumes the CSNC transaction. CSNC removes the address of the SLCB from the wait list, invokes the LOGOFF routine in DFHZCX, and then terminates.

Chapter 2.9. SNA DFC for Distributed Transaction Processing Using LU6.1

This chapter describes the SNA data flow control (DFC) for distributed transaction processing using LU6.1. Most of the control indicators used are explained in "Chapter 2.8. Intercommunication Facility Component" on page 247. If an indicator is preceded by the symbol "¬", this means that the indicator is not in the data stream. An asterisk as a character in an indicator means that any valid character can be used. For example, RQ*1 means RQD1 or RQE1, RQD* means RQD1 or RQD2.

The indicator SYNC specifies variable groups of indicators that are used for sync point. Which indicators are used depends on the context of the sync point.

Send/Receive Interface

Each CICS terminal control command in an LU6.1 session maps into the sending and/or receiving of whole chains. The LU6.1 session interface does not give the application programmer control over SNA chaining protocols. This greatly simplifies the view of the session that the application programmer needs to understand in order to control the session completely, and it also simplifies the knowledge required in order to develop a working system.

Only certain combinations of DFC indicators are valid and used by CICS LU6.1 support.

		EB		RQE1
BB		or		or
or	and	CD	and	RQD1
¬BB		or		or
		¬CD		SYNC

Figure 95. DFC Indicators for Distributed Transaction Processing

Figure 95 on page 289 shows the acceptable DFC combinations that CICS supports. The columns represent alternatives, any selection from any of the columns being valid.

Five different actions can occur on an application program interface (API) command. These actions can be viewed as operations on a buffer containing chains (DFC indicators and user data):

1. The chain in the buffer can be modified by the addition of DFC indicators – called **piggy-backing.**
2. The chain in the buffer can be sent – called **flushing.**
3. User data and DFC indicators from the command can be sent.
4. User data and DFC indicators satisfying the command can be received.
5. User data and DFC indicators from the command can be placed in the buffer – called **deferral.**

Each API command must be understood in terms of each of these five actions.

	PIGGY-BACK	FLUSH	SEND	RECEIVE	DEFER
CONV-VERSE	no	yes	yes	yes	no
SEND	no	yes	no	no	yes
SEND+WAIT	no	yes	yes	no	no
RE-CEIVE	yes	yes	no	yes	no
WAIT	no	yes	no	no	no
FREE	yes	yes	no	no	no
SYNC-POINT	yes	yes	no	no	no
RETURN	yes	yes	no	no	no

Figure 96. Actions Performed for CICS API Commands

Figure 96 on page 290 shows the effect of each API command interacting with the LU6.1 session. The rows represent the actions that can be performed on each command. The columns represent each possible API command. The entries describe which actions are performed for each command. Notice that there is redundancy in this table. All the **no-data** commands (WAIT, FREE, SYNCPOINT and RETURN) are the same, except that WAIT has nothing to piggy-back. CONVERSE and SEND with WAIT are the same, except for the receipt of data that CONVERSE implies. SEND and SEND with WAIT are the same except for the way that data send is deferred for SEND without WAIT.

The order of execution of each CICS API command is always:

1. Piggy-back (if appropriate)
2. Flush (if appropriate)
3. Send and/or receive or defer data (as appropriate).

Figure 96 on page 290 should be used to tell, for each API command, which actions are to be performed. All API requests flush data, and the only API command that leaves data deferred is a SEND without WAIT. Piggy-backing is the act of adding SNA DFC indicators to a deferred command. The command must be deferred in order for there to be anything to add DFC to. Piggy-backing is good because if the DFC is not piggy-backed, then an extra SNA flow is generated by CICS, merely to transmit the DFC. This takes the form of an SNA LUS(0006) command carrying the appropriate DFC. There is a performance degradation if unnecessary flows are generated in this way. The use of deferred commands enables CICS to optimize flows by eliminating LUS(0006) in many cases in a uniform way.

Each CICS terminal control command produces definite SNA DFC indicators (unless piggy-backing occurs). The precise indicators produced depend on the current state of the LU6.1 session.

The states of the CICS LU6.1 session are:

Begin bracket pending (BBP) state
Send (S) state
Receive (R) state
Receive pending (RP) state
Between bracket (BETB) state
Free state.

Send state is the state obtaining when this end of the LU6.1 session has received CD, or, initially, after this end has allocated the session.

Receive state is the state obtaining when the other end of the LU6.1 session has received CD.

Receive pending state is the state obtaining when a deferred request carries CD, and thus the application cannot SEND, even though the other side has not yet received CD.

Begin bracket pending state is the state obtaining between ALLOCATE and the first flow, that is, while the LU6.1 session is taken but not used.

Between bracket state is the state obtaining after sending or receiving EB.

Free state is the state obtaining after the application has freed the LU6.1 session, that is, the session is ready for use by others.

Note: For readers who are familiar with SNA DFC protocols, the above states are only approximately related to the pertinent SNA state. The reason that they are not identical is that the CICS API protects the user from much of the

complexity of SNA DFC states – the names are preserved in an effort to reduce the number of mnemonics.

	BBP	S	R
SEND	BB ¬CD RQE1	¬BB ¬CD RQE1	error
SEND INVITE	BB CD RQE1	¬BB CD RQE1	error
SEND INVITE DEFRESP	BB CD RQD1	¬BB CD RQD1	error
SEND LAST	BB EB RQE1	¬BB EB RQ*1	error
SEND LAST DEFRESP	BB EB RQD1	¬BB EB RQD1	error
CONVERSE	BB CD RQE1 receive	¬BB CD RQE1 receive	error
RECEIVE	BB CD RQE1 LUS(0006) receive	¬BB CD RQE1 LUS(0006) receive	receive
FREE	BB EB RQE1 LUS(0006) or no flow	¬BB EB RQ*1	error
SYNCPOINT	no flow	¬BB ¬CD SYNC	error
RETURN	BB EB RQE1 LUS(0006) or no flow	¬BB EB SYNC	error

Figure 97. DFC Indicators Produced by Stand-Alone API Commands

Figure 97 shows the DFC produced by each API command as a **stand-alone** operation. Stand-alone is used here to indicate that no piggy-backing is performed, either when the command is issued or after the command is issued. To find what happens if these commands are piggy-backed, see Figure 99 on page 292 to Figure 101 on page 293.

The columns represent the state that the session can be in at the time that a command is issued; the rows represent each API command. The entries represent the DFC indicators that flow with the execution of that request.

RQ*1 is used to denote RQD1 or RQE1.

	BBP	S	R	RP	BETB
WAIT	BBP	S	R	R	error
CONVERSE	n/a	n/a	error	error	error
RECEIVE	n/a	n/a	n/a	n/a	error
SEND	S	S	error	error	error
SEND WAIT	S	S	error	error	error
SEND INVITE	RP	RP	error	error	error
SEND INVITE WAIT	R	R	error	error	error
SEND LAST	BETB	BETB	error	error	error
FREE	FREE	FREE	error	error	error
SYNC– POINT	BBP	S	error	n/a	BETB
RETURN	FREE	FREE	error	error	FREE

Figure 98. States Following CICS API Commands

Figure 98 shows the states following an API command. The columns represent the state the session can be in at the time a command is issued. The rows represent each API command. The entries define the final state that the command produces on completion. The "n/a" entries indicate that the resultant state is not controlled by this side of the LU6.1 session – Figure 102 and Figure 103 on page 293 describe these events in detail. CICS LU6.1 terminal control support normally expects to be one chain behind on the session. This means that CICS maintains a buffer equal to one API command and will queue one command, if appropriate.

```
INVITE <----------> CD
LAST   <----------> EB
DEFRESP <---------> RQD1
```

	SEND	SEND DEFRESP	SEND INVITE	SEND INVITE DEFRESP	SEND LAST	SEND LAST DEFRESP
RECEIVE	CD added	CD added	no change	no change	error	error
FREE	EB added	EB added	error	error	no change	no change
SYNC-POINT	SYNC added	RQD1 -> SYNC	SYNC added	RQD1 -> SYNC	SYNC added	RQD1-> SYNC
RETURN	SYNC+EB added	RQD1 -> SYNC+EB	error	error	SYNC added	RQD1 -> SYNC

Figure 99. DFC Indicators Added to a Deferred Request

	SEND	SEND DEFRESP	SEND INVITE	SEND INVITE DEFRESP	SEND LAST	SEND LAST DEFRESP
RECEIVE	CD RQE1	CD RQD1	CD RQE1	CD RQD1	error	error
FREE	EB RQE1	EB RQD1	error	error	EB RQE1	EB RQD1
SYNC-POINT	¬CD SYNC	¬CD SYNC	CD SYNC	CD SYNC	EB SYNC	EB SYNC
RETURN	EB SYNC	EB SYNC	error	error	EB SYNC	EB SYNC

Figure 100. DFC Indicators Resulting from Piggy-Back Operation

We see from Figure 96 on page 290 that only RECEIVE, FREE, SYNCPOINT and RETURN can cause piggy-backing. Since only SEND without WAIT can produce a deferred command, we are concerned with the interaction of these four commands with the options on SEND. The relevant options on SEND are INVITE, LAST and DEFRESP. INVITE and LAST are incompatible, since they map onto CD and EB respectively.

Figure 99 shows what changes occur to the DFC indicators of a deferred flow when commands are executed that piggy-back extra indicators. The columns represent the command that is deferred at the time that a command that can be piggy-backed is issued. The rows represent an API command that can be piggy-backed. The entries represent the SNA DFC indicators that are added to the deferred request. The entry RQD1 → SYNC denotes the replacement of an RQD1 indicator by a SYNC indicator.

Figure 100 shows the DFC indicators that result from a piggy-back operation. The columns represent the command that is deferred at the time that a command that can be piggy-backed is issued. The rows represent an API command that can be piggy-backed. The entries represent the DFC indicators that result from the piggy-back operation.

	SEND	SEND INVITE	SEND LAST
RECEIVE	n/a	n/a	error
FREE	BETB	error	BETB
SYNC-POINT	S	n/a	BETB
RETURN	FREE	error	FREE

Figure 101. States Following a Piggy-Back Operation

Figure 101 shows the state of the LU6.1 session that results from a piggy-back operation. The columns represent the deferred API command; the rows represent the action to be piggy-backed. The entries represent the final state reached. The "n/a" entries indicate that the resultant state is not controlled by this side of the LU6.1 session.

API commands that result in the receipt of inbound data (that is, RECEIVE and CONVERSE) leave the state of the LU6.1 session undefined. The reason for this is that session state is dictated by the inbound DFC indicators. Because the receipt of data carries with it the possibility of receiving DFC indicators and thus having future actions constrained, being in receive state is often associated with losing control. The act of sending CD (that is, being prepared to receive DFC indicators) is synonymous with giving up control. Figure 97 on page 291 shows that there are many more actions possible in the send state rather than in the receive state.

If an LU6.1 session is in receive state, then RECEIVEs only can be issued – all other commands result in error. The result of a RECEIVE command is to receive not only data, but also DFC indicators.

The relevant DFC indicators are CD or EB with or without SYNC.

If CD is not received, then the session remains in the receive state.

If CD is received, then the session is in the send state.

If EB is received, then the session is in the BETB state.

If SYNC is received, a SYNCPOINT or RETURN is required.

	¬CD	CD	EB	¬CD + SYNC	CD + SYNC	EB + SYNC
DFHFREE	off	off	on	off	off	on
DFHSYNC	off	off	off	on	on	on
DFHRECV	on	off	off	on	off	off

Figure 102. EIB Indicators after the RECEIVE or CONVERSE Command

Figure 102 shows the information that a RECEIVE or CONVERSE returns. The columns represent the DFC indicators received. The rows represent the EIB flags set on completion of the RECEIVE or CONVERSE command. The entries define the EIB setting in each case.

¬CD	CD	EB
R	S	BETB

Figure 103. State of LU6.1 Session after RECEIVE or CONVERSE Command

Figure 103 shows the state that the LU6.1 session is in after a RECEIVE or CONVERSE command, and a SYNC is not received. The columns represent the DFC indicators received and the state resulting from the receipt of that indicator.

If a SYNC indicator is received, then the receiver must take a sync point through execution of a SYNCPOINT or RETURN API command or through execution of a DL/I TERM call.

	¬CD + SYNC	CD + SYNC	EB + SYNC
SYNCPOINT	R	S	BETB
RETURN	error	FREE	FREE

Figure 104. State Following Receipt of SYNC Indicator

Figure 104 shows the state that the LU6.1 session is in after the receipt of a SYNC indicator and after the appropriate API command has been issued. The columns represent the DFC indicators received. The rows represent the two valid API commands that can then be issued. The entries describe the subsequent state in each case.

The method used by CICS to sync point depends on whether one or more mirror tasks are active. When only one mirror task is active, then a sync point request is sent and a response is received, involving two flows. When more than one mirror task is active, then all but one of the mirrors receive a PREPARE request before sending a sync point request, thus requiring three flows altogether.

The following figures give the flows for both types of sync point associated with three different DFC indicators. In each figure, the first entry is with deferred data, and the second entry is without deferred data.

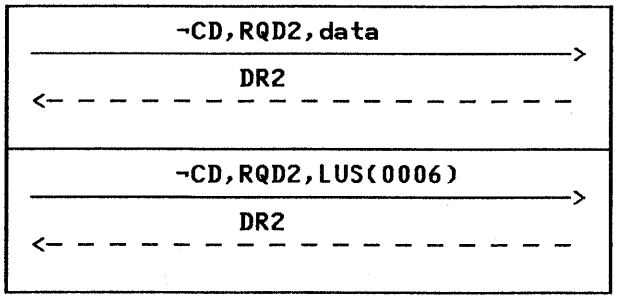


Figure 105. Two-Flow Sync Point, ¬CD

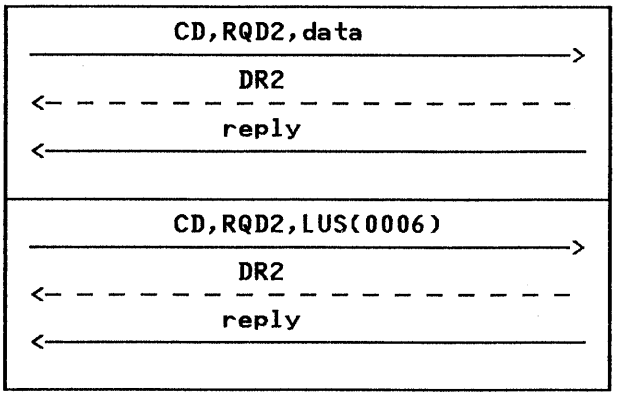


Figure 106. Two-Flow Sync Point, CD (Not Optimized)

CICS sends and receives this not optimized version of sync point.

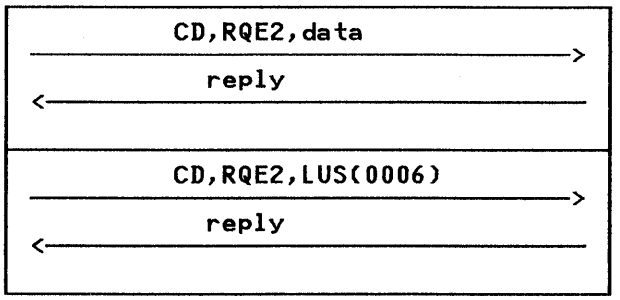


Figure 107. Two-Flow Sync Point, CD (Optimized)

CICS can receive these flows, but not send them. (That is, CICS can be on the right hand side of this figure, but not the left.) Other LU6.1 subsystems, however, do support this flow.

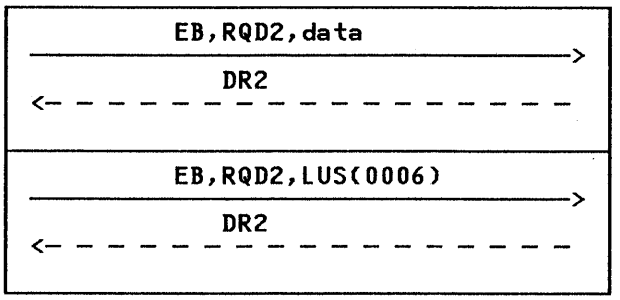


Figure 108. Two-Flow Sync Point, EB

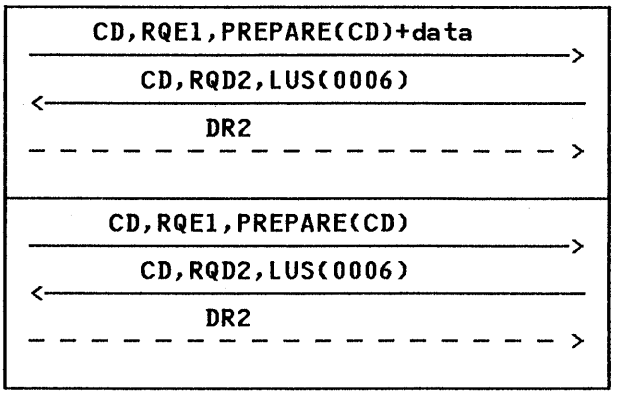


Figure 109. Three-Flow Sync Point, ¬CD

PREPARE(CD) is the PREPARE command that insists that CD is returned on the SPR flow.

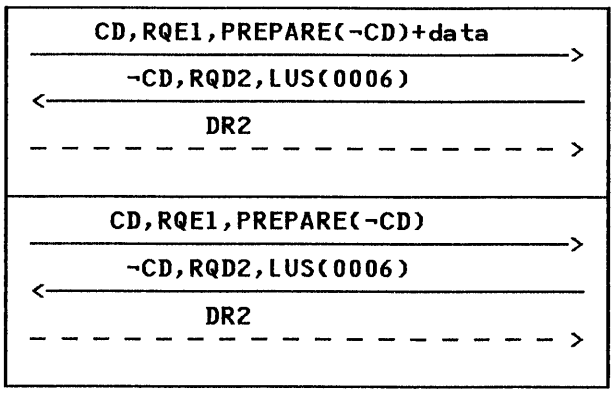


Figure 110. Three-Flow Sync Point, CD

PREPARE(¬CD) is the PREPARE command that does not force CD to be returned.

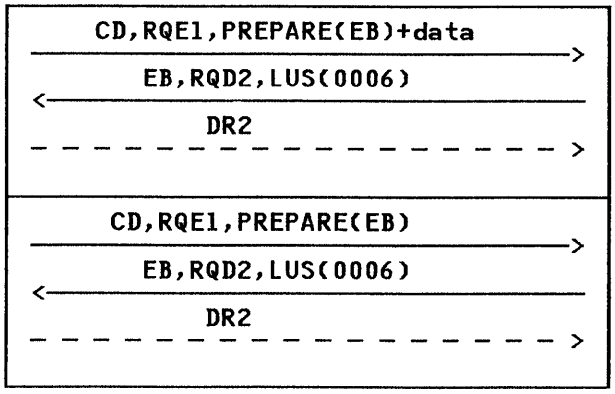


Figure 111. Three-Flow Sync Point, EB

The application can be in conversation with more than one LU6.1 session. Each session is accessed explicitly using session-local terminal control commands, implicitly using SYNCPOINT and RETURN, or implicitly using function shipping. When two applications are in conversation, then either of them is able to issue a SYNCPOINT command. CICS does not support the arbitrary initiation of sync points in this way; the application in the send state is the one that can initiate sync point. This is consistent with the idea that the application in the send state is in control. The application that receives the SYNC indicator must react to it by issuing a SYNCPOINT or RETURN in reply.

When multiple conversations are taking place, the initiator of a sync point must be in the send state with all of its LU6.1 sessions. An application with multiple conversations that receives a SYNC indicator must, before replying with a SYNCPOINT or RETURN command, be in the send or the BETB state with the rest of its conversations (that is, the conversation from which the SYNC indicator was received is discounted, but the application must get into the right state with all the other conversations). This is less complex than it seems. If an application has multiple conversations, then it will typically be in the send state with all except one conversation. This is because one can wait on (RECEIVE from) only one conversation at a time. For example, with function shipping conversations, the application is never in any state other than the send state – the receive states that occur are always hidden by function shipping logic. In fact, in any conversation where CONVERSE is the only command used, the application always sees just the send state.

Given that no application ever executes with more then one conversation not in the send state, we see the idea of control in a more general context. CICS permits applications to acquire (ALLOCATE) conversations to multiple applications. These in turn can then acquire new conversations. The total structure forms a tree. Each application represents a node in the tree, and each conversation represents a branch. The structure is a tree because loops are prevented due to the fact that CICS provides no mechanism for starting a conversation with an already executing application.

The application in the send state with all its conversations is in control of the whole tree. If an application passes CD to another application, then it gives up control to that application. Applications that are not in control are in the receive state with exactly one conversation – the **master**. The conversations that it is in the send state with are connected to applications that are **slaves** to that **master**. Thus the whole tree can be expressed as a downward progression of **master-slave** relationships.

The sync point protocol ripples out in waves from the application in control down to the **slaves**. They, in turn, become the **masters** to their **slaves**. It is important to note that the whole **master-slave** relationship is dynamic – at any time CD can be sent from the application in control to a neighboring application. The application receiving

CD is then in control – the application that used to be in control is now the **slave**.

This ordered system can be upset by misuse of the CICS API. Use of the SEND INVITE command enables an application to give up control to more than one application. These then look like competitors for control of the tree. In such cases it is entirely the responsibility of the application programmer to ensure that the competitors do not all attempt to initiate a sync point. The error is detected by CICS at some application that is unfortunate enough to receive the conflicting SYNCPOINT requests. If SEND INVITE is not used, the situation does not arise.

Consider an application connected to an LU6.1 session. Errors can occur in three places:

- In logic unrelated to the session
- In and around the terminal control API
- In the LU6.1 session itself.

An application can, by using HANDLE ABEND, be able to recover from errors 1 and 2 and continue that conversation normally. This is not the case with error 3. The LU6.1 session has failed and the conversation has terminated. There is nothing the application can do to bring back either the session or the conversation. The application is informed of the error by being abended with abend code ATNI or ASP3.

ASP3 is the abend code used to indicate that the error was received by the sync point program (DFHSPP).

ATNI is the abend code that is used at other times.

HANDLE ABEND can be invoked, but the application is not be able to access the session – any attempt to do anything other than a FREE, SYNCPOINT, or RETURN affecting that session will result in a further abend. The effect of FREE is to release the dead session. The effect of SYNCPOINT is to ignore the session completely; FREE is still required. The effect of RETURN is to issue an implicit FREE against the failing session.

Consider now two applications that are in conversation using an LU6.1 session. One of the applications fails. CICS takes responsibility, whichever end of the LU6.1 session that it is executing on, for cleaning up the application-session connections. CICS uses LU6.1 error recovery protocol (ERP) architecture to propagate the error to the other end of the session, and then uses the same architecture to terminate the SNA bracket.

The application causing the failure is of no further interest in this discussion – it has gone away. The application left behind, however, is about to be hit by news of the failure, and the following describes what happens.

The application is not always informed of the error immediately; when the error indication is received is discussed later. The application is informed of the error by being abended with abend code ATNI or ASP3. These are the same abend codes as for session error. Thus the application is told that the conversation has failed – but not how.

HANDLE ABEND can be invoked but the application will not be able to access the session – any attempt to do anything other than a FREE, SYNCPOINT or RETURN affecting that session will result in further abend. The effect of FREE is to release the dead session. The effect of SYNCPOINT is to ignore the session completely – FREE is still required. The effect of RETURN is to issue an implicit FREE against the failing session.

CICS takes responsibility for the termination of the SNA bracket; an installation must code a node error program if it wishes to do something different from the CICS default actions. If this is done, then any change to session flows should be made in accordance with SNA LU6.1 architecture, otherwise a private protocol has been formed.

The description of which flows CICS produces to clean up the session is not pertinent to this discussion; the application programmer must understand, however, that in this error state, data may have been lost. The application design must rely upon SYNCPOINT to recover from this sort of error if necessary. If confirmation of delivery is essential, DEFRESP can be used. In inquiry applications, the loss of data is immaterial, and reentering the inquiry is appropriate.

The time at which the application program receives the error is now described.

First, session errors are received for any command affecting that session (SYNCPOINT, RETURN, and ALLOCATE included).

Second, application errors propagated over the session are accepted at the time that an inbound flow occurs for:

RECEIVE
CONVERSE
SYNCPOINT
RETURN (if not BETB)
FREE (sometimes)

or when a DEFRESP is waited for.

DEFRESP is waited for when an application either issues a SEND WAIT DEFRESP or a WAIT after a deferred SEND DEFRESP. If a SEND DEFRESP is deferred (because it has no WAIT on it), then the next API command affecting that session will have an implied WAIT and so will accept the error.

The FREE command causes errors to be accepted sometimes. CICS makes a performance/simplicity trade-off that does not force a definite response with every EB. The reason that EB,RQD1 is good is that the application gets all its errors delivered to it - so it is easy to tell where the error occurred, and the dump and trace contain useful information. The disadvantage of EB,RQD1 is that the application must wait for the DR1 response. This consumes resource (real storage and the LU6.1 session).

CICS puts RQD1 with an EB flow when it is likely to be useful. In practice, if an application asks a question and sends CD, then the answer can often come back EB. It is of little interest to the application answering the question that the answer was not liked. If, however, an application is driving another application, then it will be sending many flows. If it indicates completion by SEND LAST or FREE, then CICS places RQD1 with the EB to collect the errors.

If the EB indicator is the only one since a flow with CD, then the EB goes with RQE1 (exception response mode). If the EB indicator is the last of more than one, then the EB goes with RQD1 (definite response mode).

Thus FREE causes errors to be accepted if EB goes with RQD1; when the EB goes with RQE1, errors can not be delivered to the application. The application can ensure that it receives all its errors by issuing SEND LAST DEFRESP WAIT or a RETURN (thus entering sync point logic).

Normal Flows without ERP

In the following figures, question marks in the flow mean that the other system can send any message to the CICS system.

Starting the Conversation

A set of indicators that are enclosed in parentheses is the set that is generated before piggy-backing takes place.

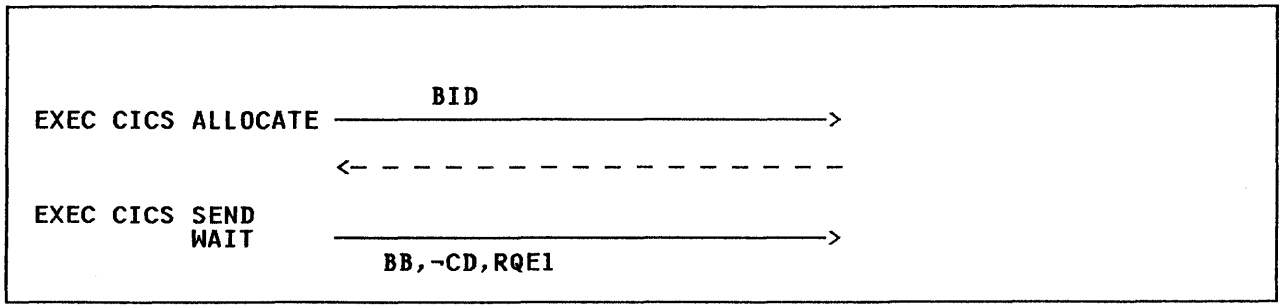


Figure 112. CICS Is Primary, SEND in BBP State

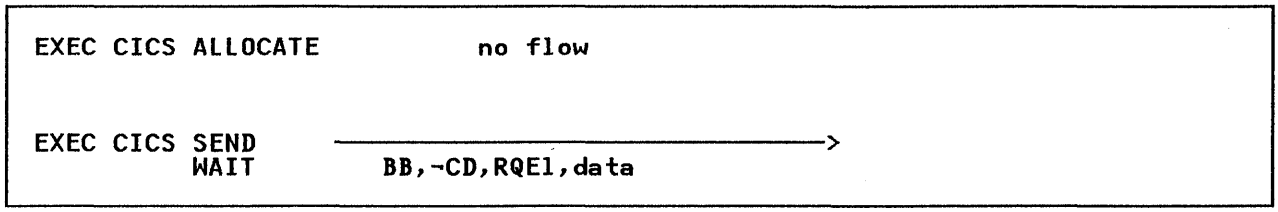


Figure 113. CICS Is Secondary, SEND in BBP State

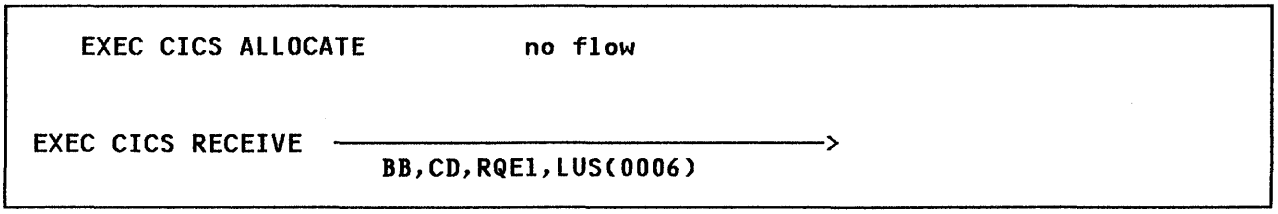


Figure 114. CICS Is Secondary, RECEIVE in BBP State

Simple Requests

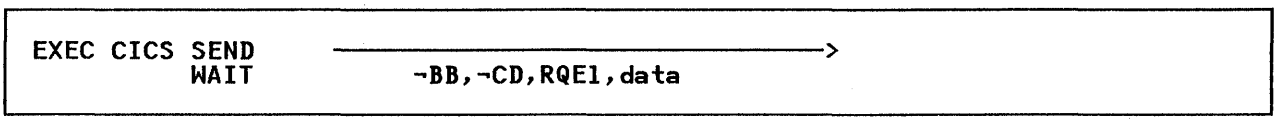


Figure 115. SEND in Send State

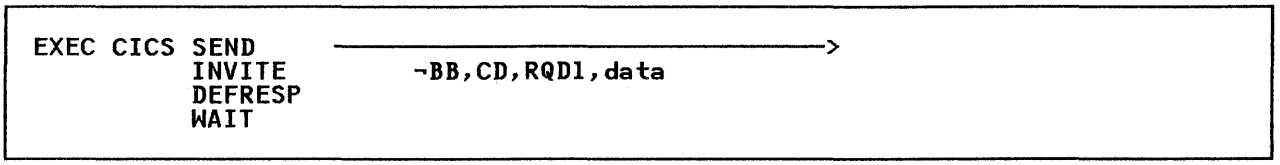


Figure 116. SEND INVITE DEFRESP in Send State

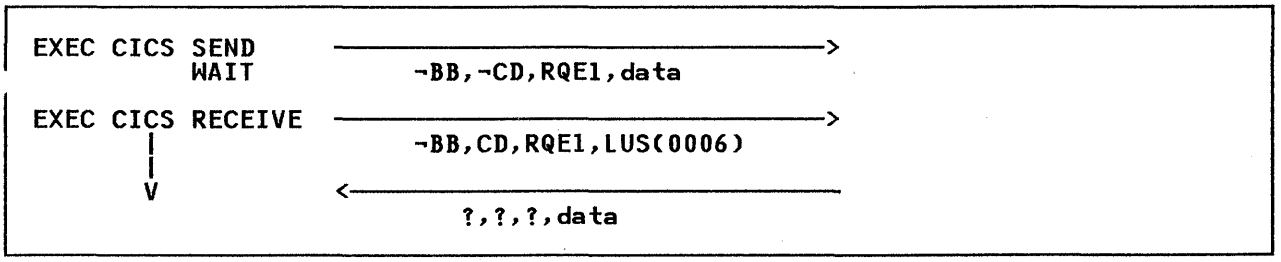


Figure 117. RECEIVE in Send State

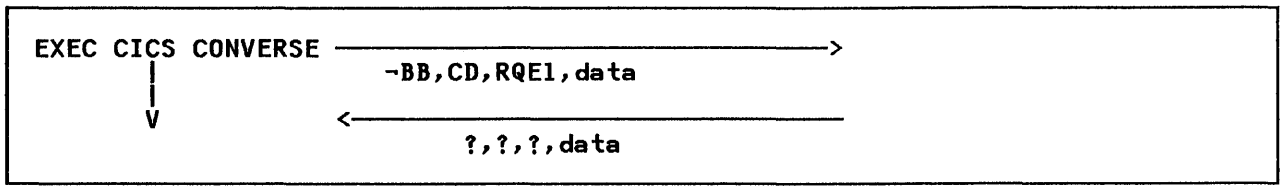


Figure 118. CONVERSE in Send State

Piggy-Backing Requests

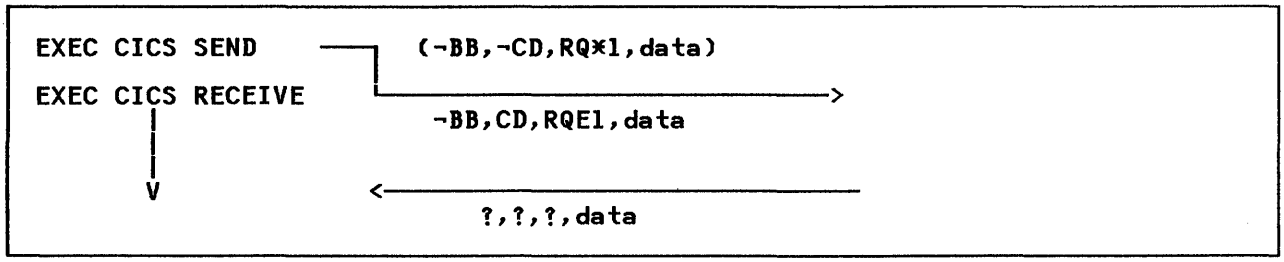


Figure 119. RECEIVE after Deferred SEND

Flushing Deferred Flows

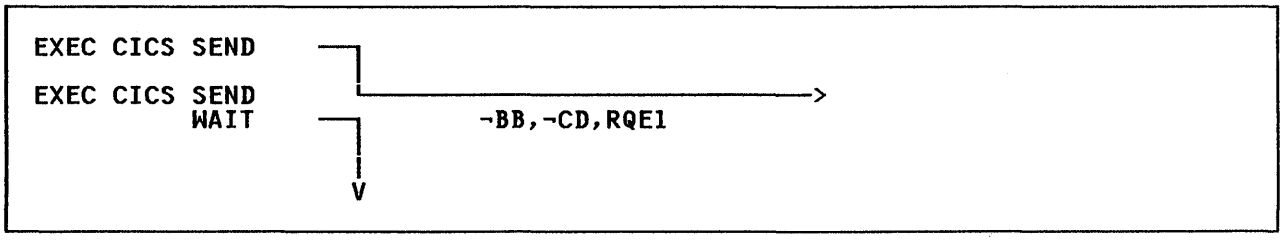


Figure 120. SEND after Deferred SEND

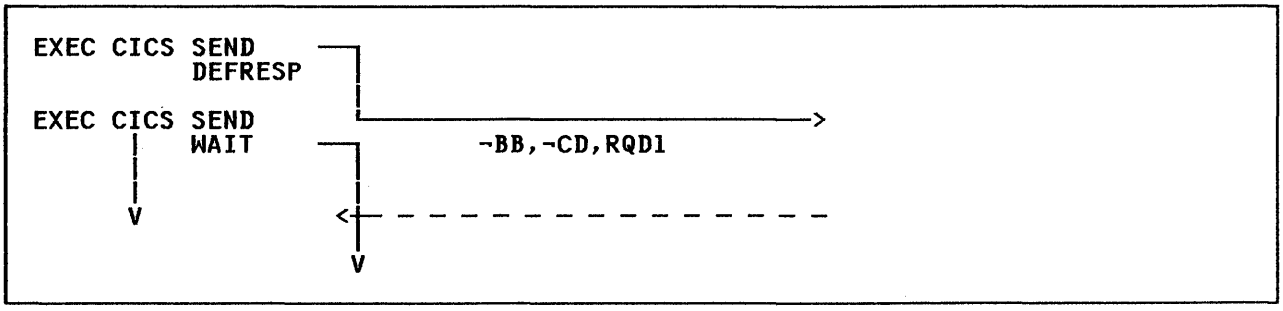


Figure 121. SEND after SEND DEFRESP

Sync Point Flows

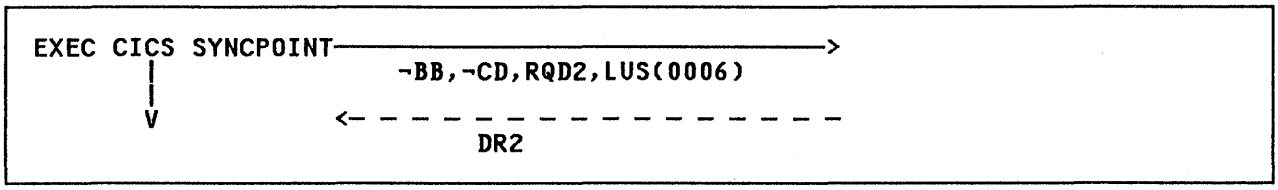


Figure 122. Two-Flow Sync Point without Deferred Data

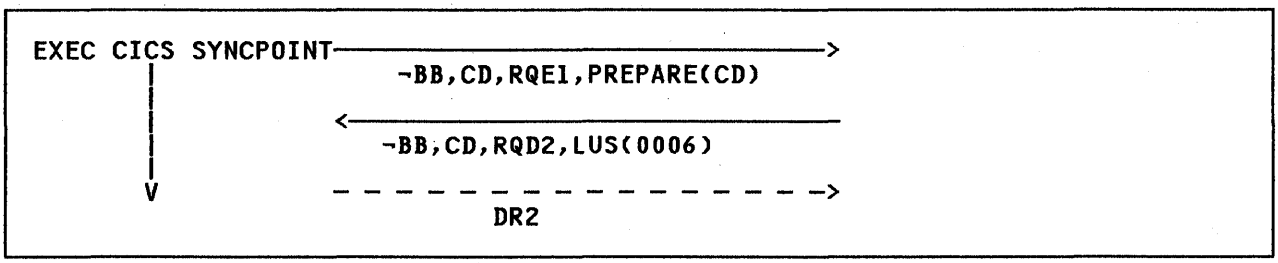


Figure 123. Three-Flow Sync Point without Deferred Data

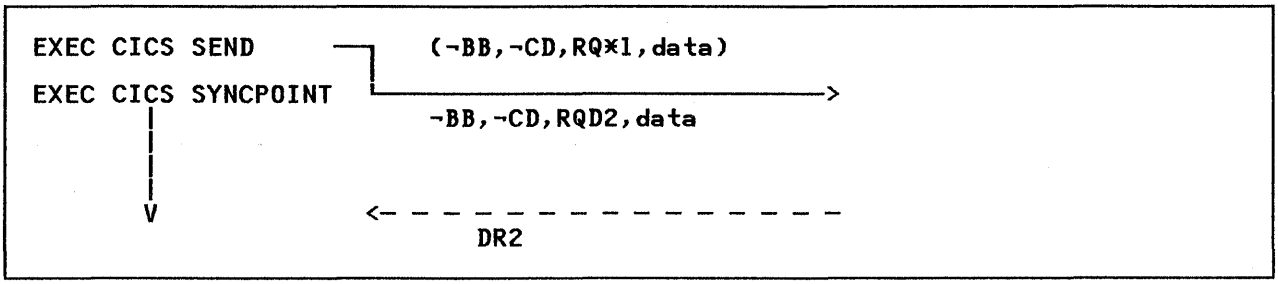


Figure 124. Two-Flow Sync Point with Deferred SEND

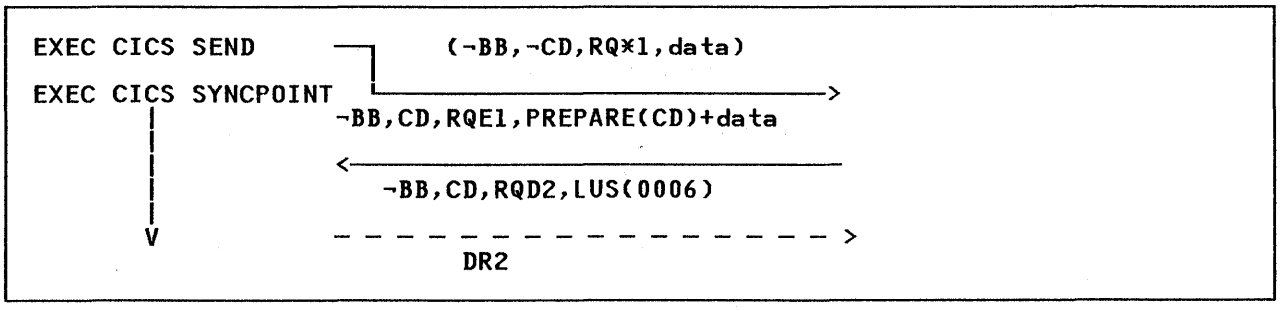


Figure 125. Three-Flow Sync Point with Deferred SEND

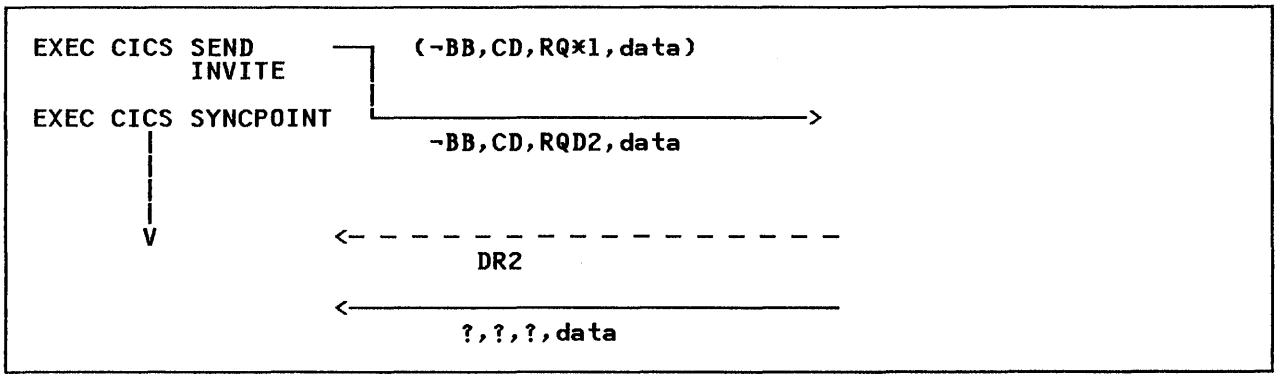


Figure 126. Two-Flow Sync Point with Deferred SEND INVITE

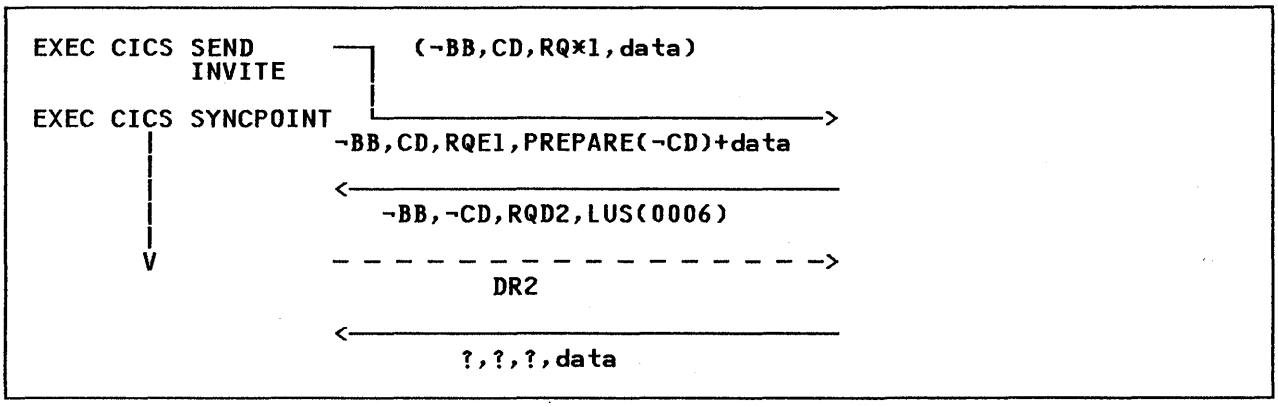


Figure 127. Three-Flow Sync Point with Deferred SEND INVITE

Terminating the Conversation without Sync Point

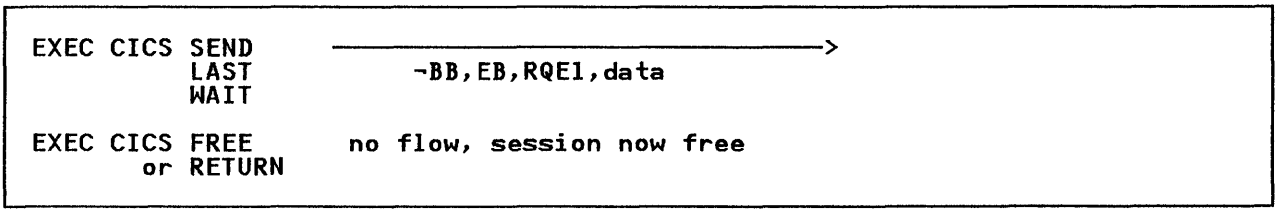


Figure 128. SEND LAST Followed by FREE or RETURN

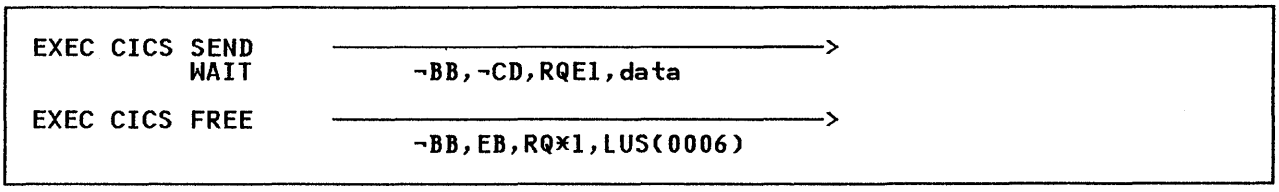


Figure 129. SEND WAIT Followed by FREE

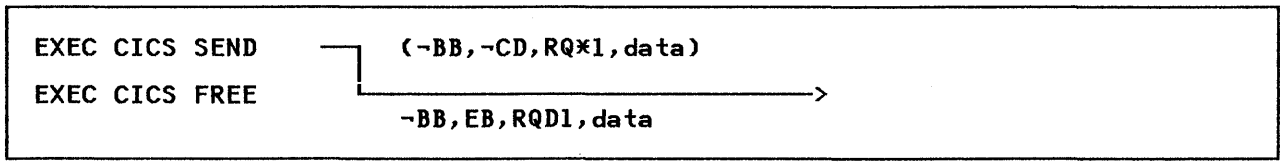


Figure 130. SEND Followed by FREE

Terminating the Conversation with Sync Point

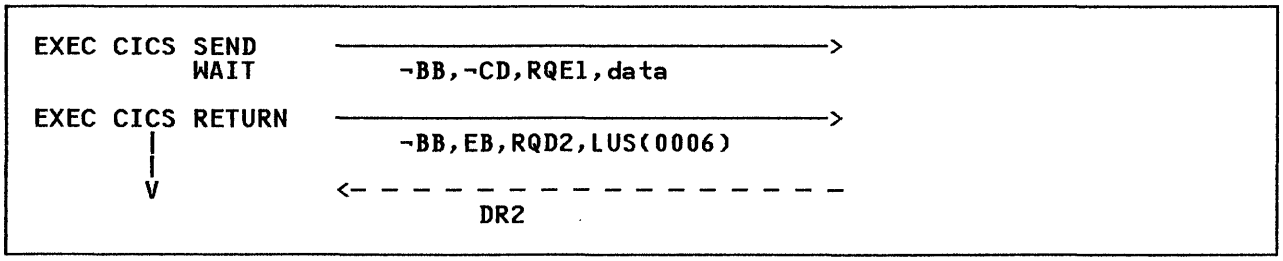


Figure 131. Two-Flow SEND WAIT Followed by RETURN

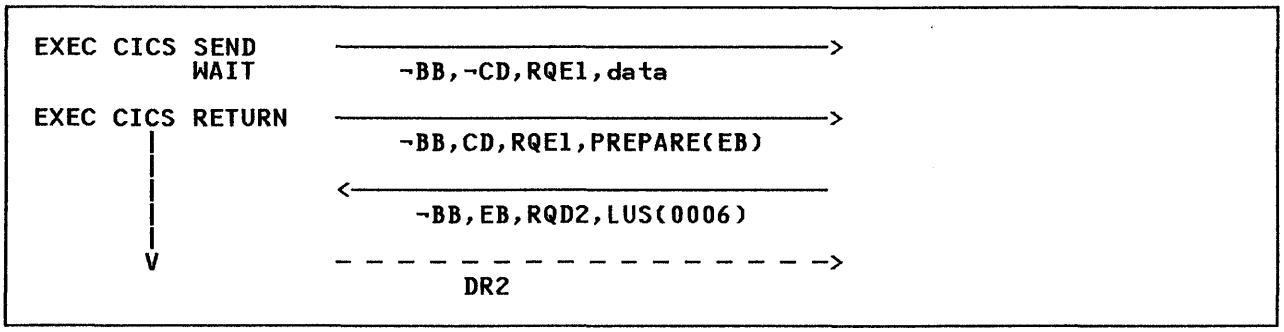


Figure 132. Three-Flow SEND WAIT Followed by RETURN

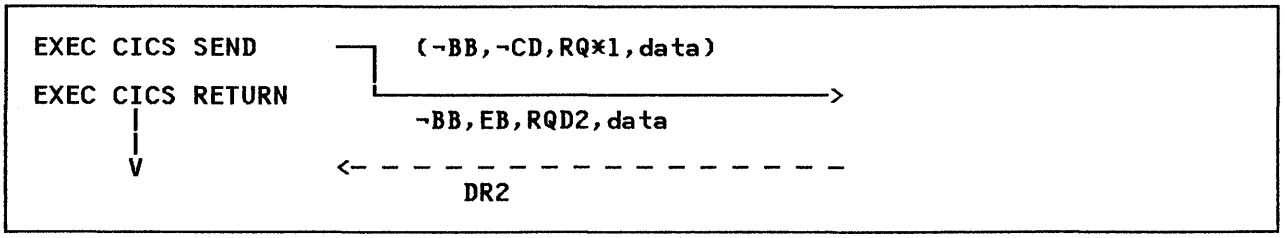


Figure 133. Two-Flow SEND Followed by RETURN

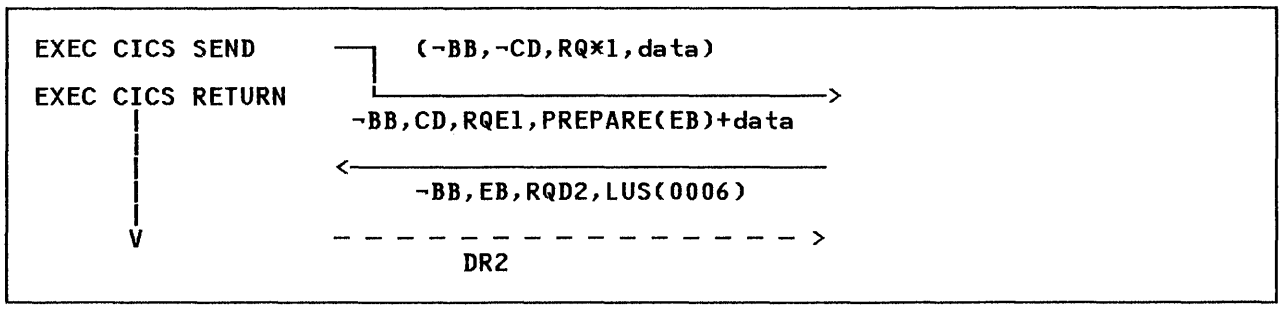


Figure 134. Three-Flow SEND Followed by RETURN

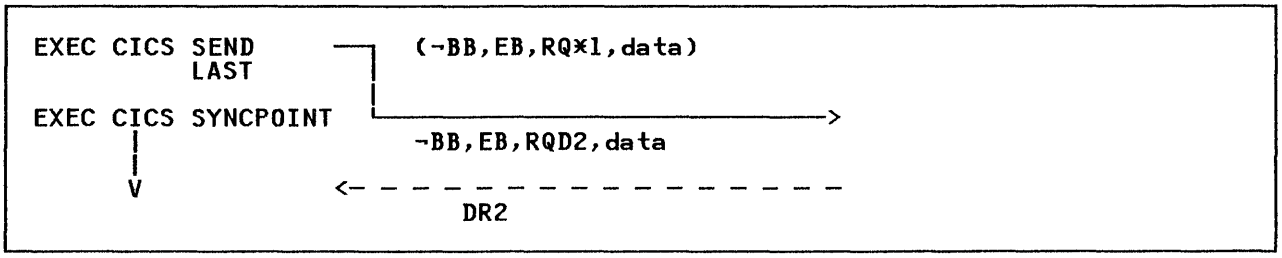


Figure 135. Two-Flow SEND LAST Followed by SYNCPOINT

START/RETRIEVE Interface

CICS uses the EXEC CICS START API to produce LU6.1 scheduler model flows. These LU6.1 requests represent unrelated messages and may be exactly one SNA chain.

CICS produces DFC indicators depending upon the sequence of session events that the application logic encodes. This is performed by holding one message in a buffer.

The LU6.1 architecture distinguishes between a PROTECTED schedule request and an UNPROTECTED schedule request. CICS makes use of this knowledge to keep track of whether an LU6.1 session that uses only SCHEDULE requests has used only UNPROTECTED SCHEDULE requests. If this is so, then the sync point protocols are redundant, and CICS deletes them. If a single PROTECTED SCHEDULE request is issued, then the sync point protocol must be entered.

Figure 136 shows the mapping from START commands to SEND/RECEIVE commands. START flows map into SCHEDULE FMH flows, CICS frees the session at SYNCPOINT time and does not perform a SYNCPOINT (freeing the session instead) if all the STARTs for a conversation are not PROTECTED.

EXEC START command	EXEC TC command issued against the link
START	CONVERSE
START NOCHECK	SEND

Figure 136. Mapping from START Commands to SEND/RECEIVE Commands

Normal Scheduler Model Flows

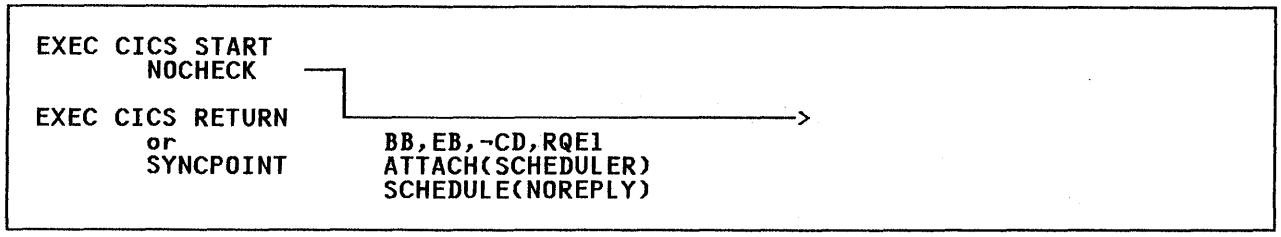


Figure 137. Single UNPROTECTED SCHEDULE Requests

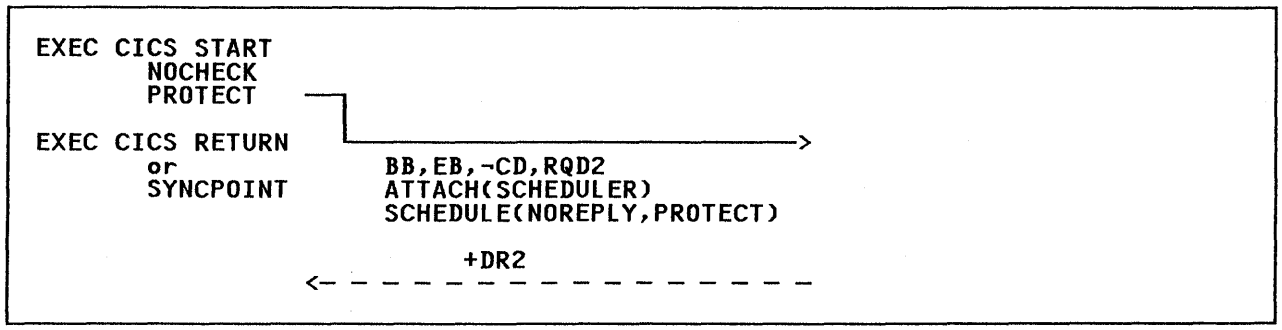


Figure 138. Single PROTECTED SCHEDULE Requests

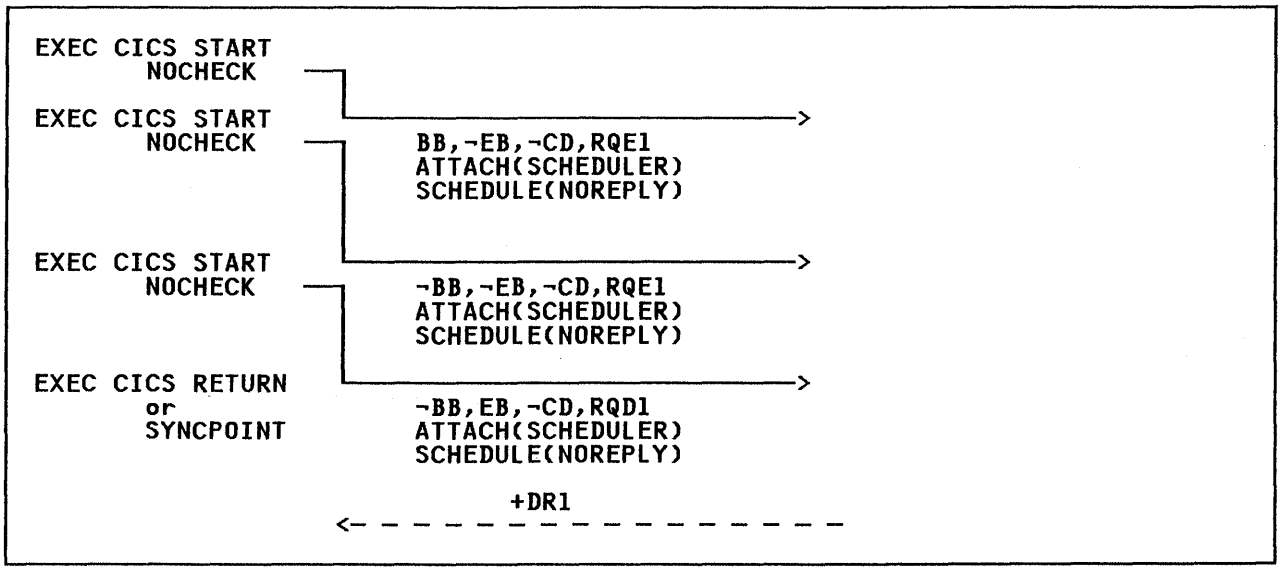


Figure 139. Multiple UNPROTECTED SCHEDULE Requests

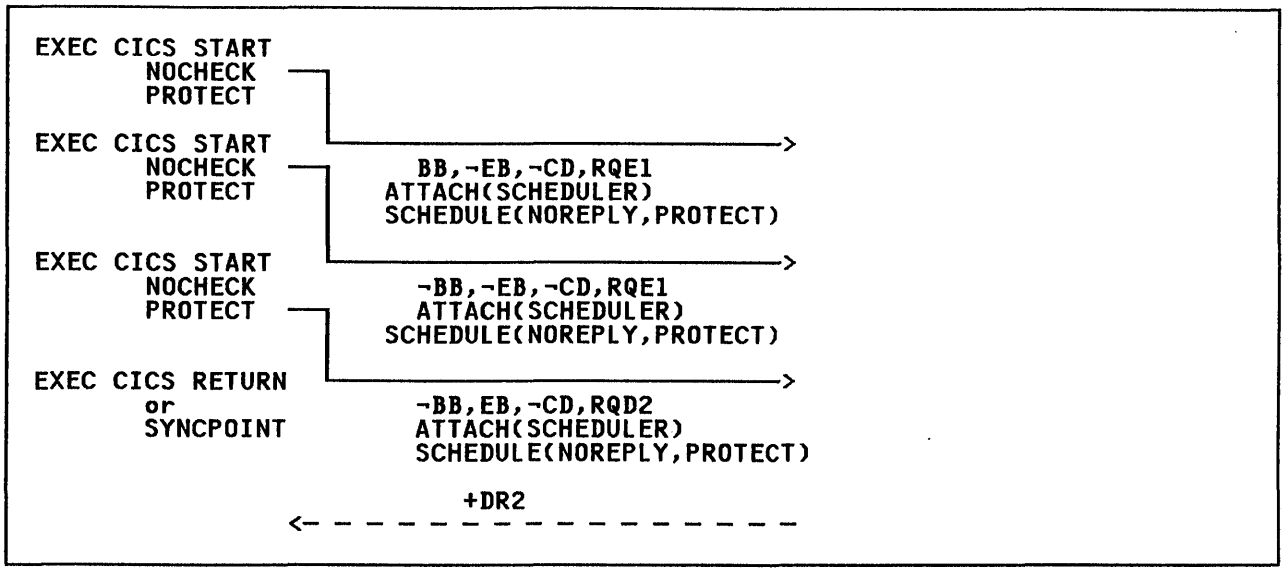


Figure 140. Multiple PROTECTED SCHEDULE Requests

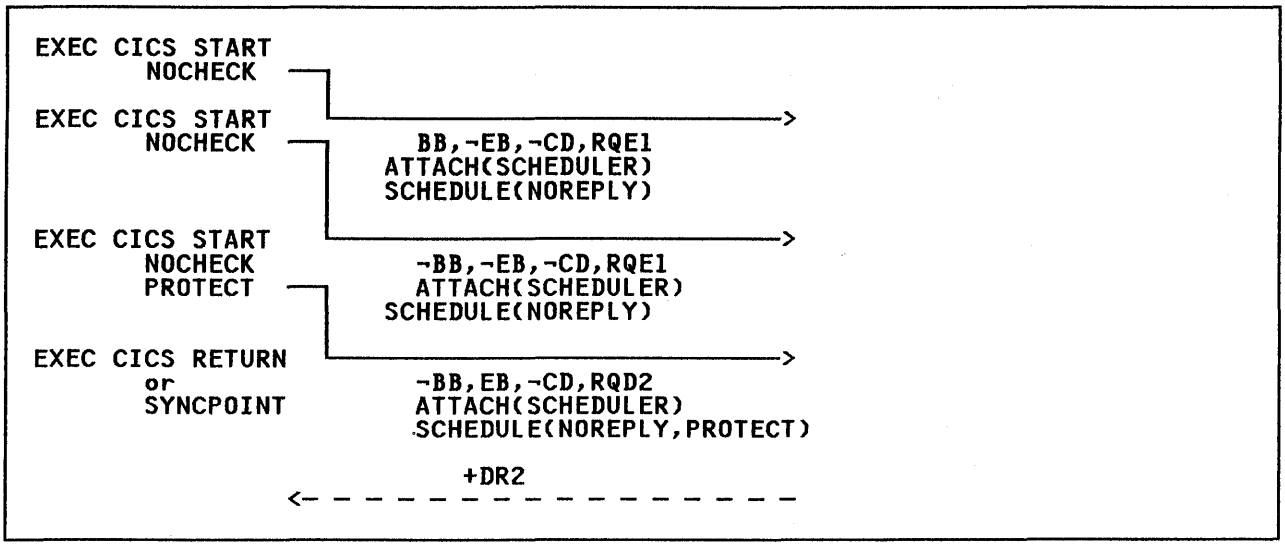


Figure 141 (Part 1 of 3). Mixed PROTECTED and UNPROTECTED SCHEDULE Requests

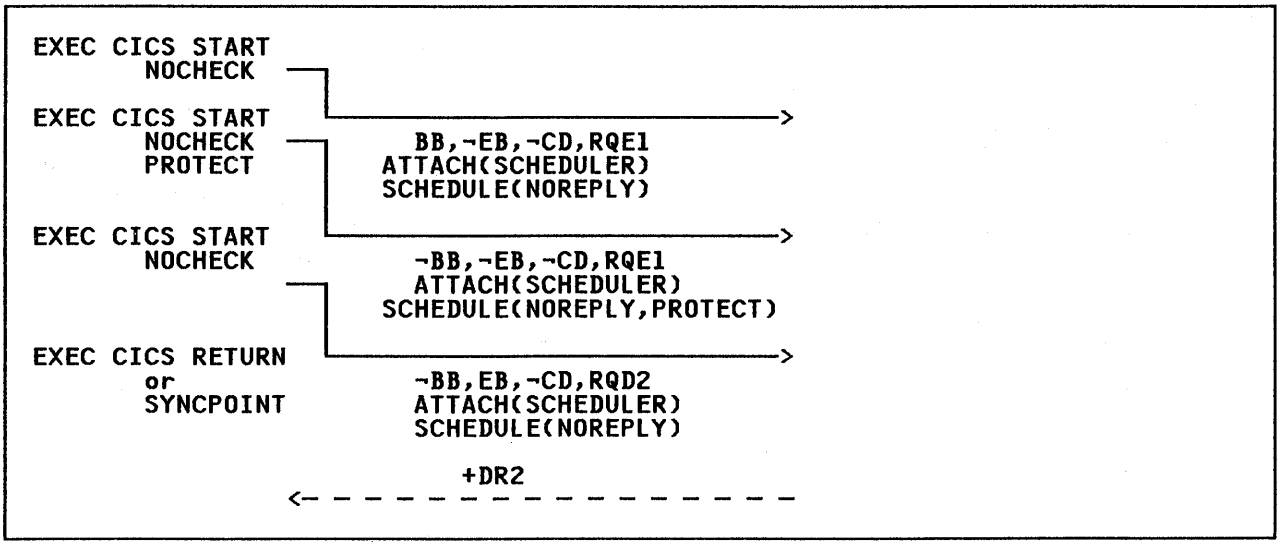


Figure 141 (Part 2 of 3). Mixed PROTECTED and UNPROTECTED SCHEDULE Requests

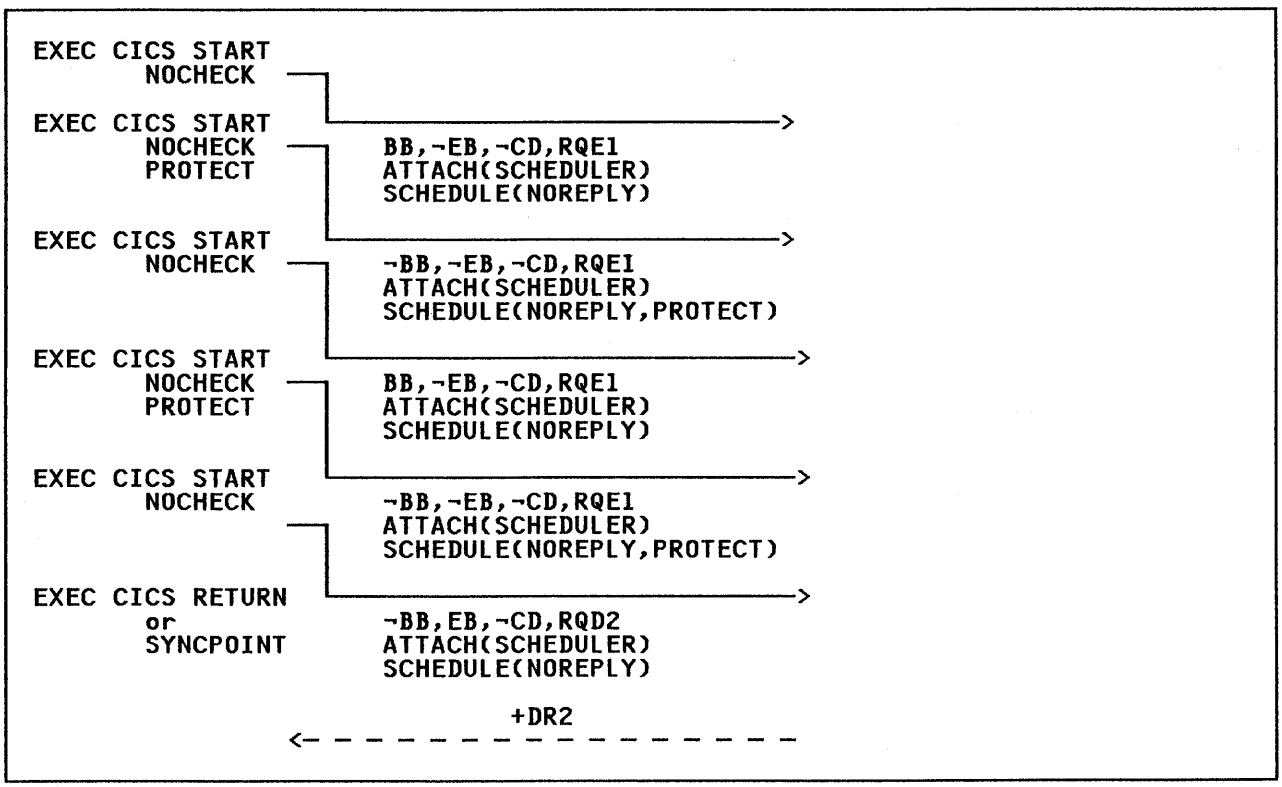


Figure 141 (Part 3 of 3). Mixed PROTECTED and UNPROTECTED SCHEDULE Requests

Notes:

1. *The messages transmitted to or from CICS are unrelated. Both the ATTACH FMH and the SCHEDULE FMH are constructed by the CICS START routine.*

2. *The messages produced are single-chain only.*

3. *The data passed by the application issuing the EXEC CICS START command can contain FMHs. This is indicated by the use of the FMH keyword on START.*

4. *The keywords NOCHECK and PROTECT on the START command are reflected in the SCHEDULE FMH.*

DFC within the CICS System

Session Setup

CICS Session Resource Qualifier Use

CICS maintains as part of its system generation the following information (bit TCTEIZRY reflecting the SEND or RECEIVE specification in the TCTTE definition for the session):

1. A static indication of whether this session is primary or secondary.

2. For a primary session, the primary session resource qualifier (PSRQ), that is the TRMIDNT of the TCTTE for the session.

3. For a secondary session, the secondary session resource qualifier (SSRQ), that is the TRMIDNT of the TCTTE for the session.

4. Optionally, the other half of the session qualifier, that is the NETNAMQ of the TCTTE definition for the session.

CICS uses the following rules for session resource qualifier (SRQ) bind exchange:

1. The system issuing REQSESS or OPNDST need not specify an SRQ. However, CICS itself always specifies its own SRQ.

2. If the system issuing the initial REQSESS or OPNDST specifies its own SRQ, then this cannot be altered by negotiation since it is, by definition, the identifier of the session.

3. If the system issuing the initial REQSESS or OPNDST specifies its own SRQ, then CICS uses this name to identify the session being opened. CICS can then return its own name altered – as primary in BIND, as secondary in BIND response.

4. If CICS receives an altered name as indicated in note 3, then CICS accepts the alteration and remembers the updated name pending a possible future session restart.

Session Resynchronization

STSN Negotiation

CICS sets the **STSN-not-required** (or **BIS-sent**) and **sequence-numbers-available** bits in the negotiable bind and response exchanges as follows:

1. CICS restart, not emergency restart. CICS sets **STSN-not-required, sequence-numbers-not-available.**

2. Session restart, not a CICS system restart:

   a. Previous session ended normally. CICS sets **STSN-not-required, sequence-numbers-available.**

   b. Previous session ended abnormally, but not while a response to an outbound RQD2 flow was outstanding – as in case 2a.

   c. Previous session ended abnormally, and a response to an outbound RQD2 flow was outstanding. CICS sets **STSN-required, sequence-numbers-available.**

3. CICS emergency restart. Those sessions (and only those) on which RQD2 had been sent or received at some time during the previous CICS run are restarted as **sequence-numbers-available**. Those (and only those) for which a response to RQD2 was outstanding are restarted as **STSN-required.**

STSN Rules

A CICS primary follows the BIND by STSN if and only if the primary or secondary (or both) set **STSN-required**. If the CICS primary has no sequence numbers available, then it uses zero for the inbound and outbound numbers. A CICS primary always uses the **TEST-and-SET** option on both flows for its first STSN and the **SET** option for the second STSN, if any (see later). A CICS secondary that receives STSN, but has no sequence numbers available, replies **RESET** on both the inbound and outbound flows.

A CICS primary follows an STSN exchange (involving one or two STSNs) with start-data-traffic (SDT), regardless of the outcome of the STSNs. A CICS secondary positively responds to SDT regardless of the outcome of the STSN exchange. It does, however, negatively respond to SDT if a required STSN was not sent.

The sequence numbers that CICS uses are those of the last RQD2 flow received and processed for the inbound and the outbound, one of the following:

1. The last RQD2 sent and the corresponding response was received.

2. The last RQD2 sent, and the session failed before the response was received, and DTB = YES or DTB = (YES,WAIT) (see note later) is in effect.

3. The penultimate RQD2 sent when the session failed after the last RQD2 was sent and its response was not received, and DTB = (YES,NO) applies.

Note: The DTB parameter, specified in the PCT for the transaction, determines the action taken by CICS in the event of a session failure after sending RQD2:

DTB = YES	Backout
DTB = (YES,WAIT)	Wait for session recovery and follow the other system
DTB = (YES,NO)	Commit

In case 2 (with DTB = YES) and case 3 in the above list, it is possible that the action of CICS can leave the session out of sync without there actually being a recovery error. In these cases a **test-negative** response to STSN drives a second STSN from a CICS primary with outbound number corresponding to the alternative RQD2. This allows the secondary to know what is going on.

STSN Flows

The following figures illustrate the rules laid out in the previous sections.

For the CICS primary cases, only the expected responses to STSN are shown. Unexpected ones produce an operator message. The next flow on the session is SDT.

For the CICS secondary cases, only the expected STSN numbers are shown. Unexpected ones are responded to as invalid on both flows and an operator message is produced.

For the BIND and BIND response flows, the only settings shown are those of the **sequence-numbers-available** and **STSN-required** bits. Where the value does not matter, **any** is written.

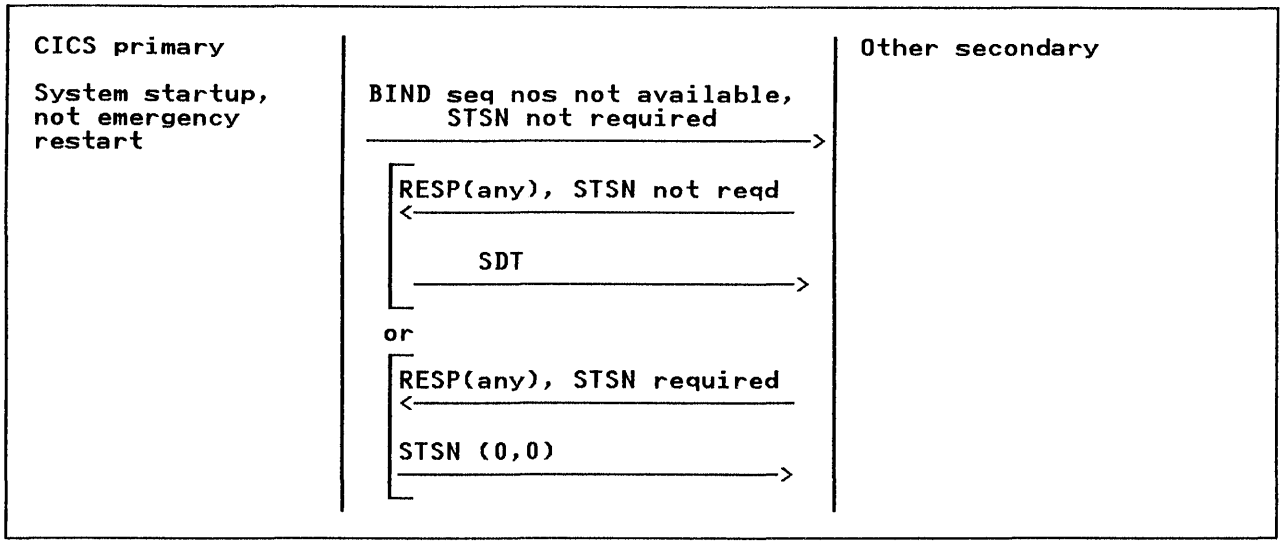


Figure 142. STSN Flows for CICS as Primary

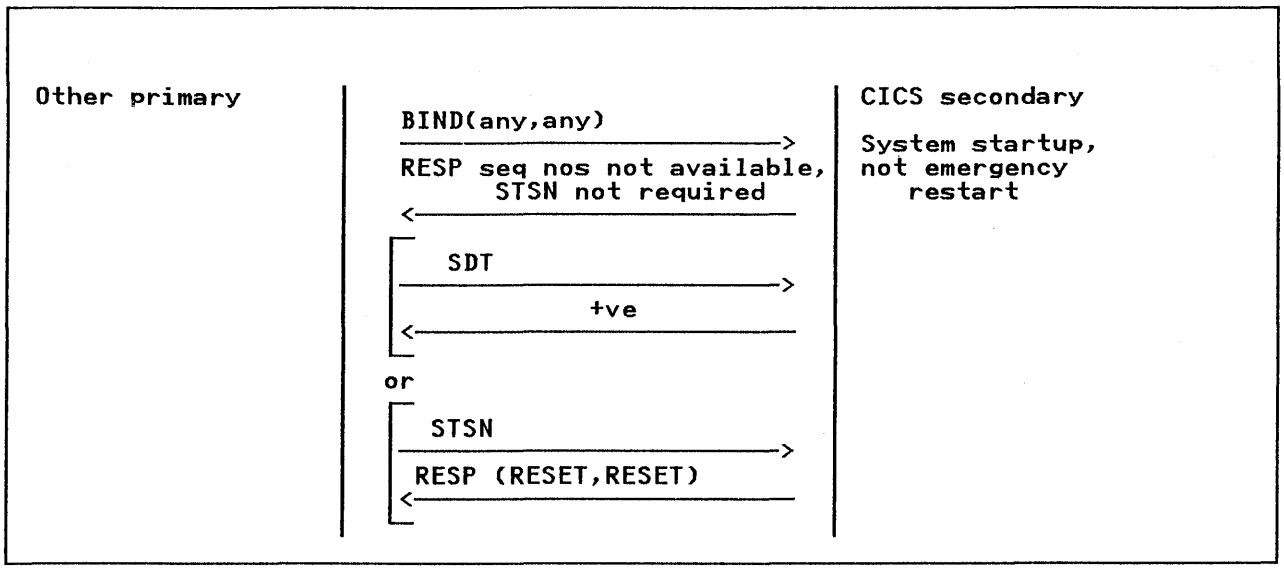


Figure 143. STSN Flows for CICS as Secondary

Figure 145 shows flows at session initiation for:

1. CICS as primary.
2. CICS has memory of the previous use of the session, that is the session ended normally or abnormally during the current CICS run, or there has been an intervening CICS failure and emergency restart.
3. The previous session ended at points (1) or (2) in Figure 144 (the only distinct cases).

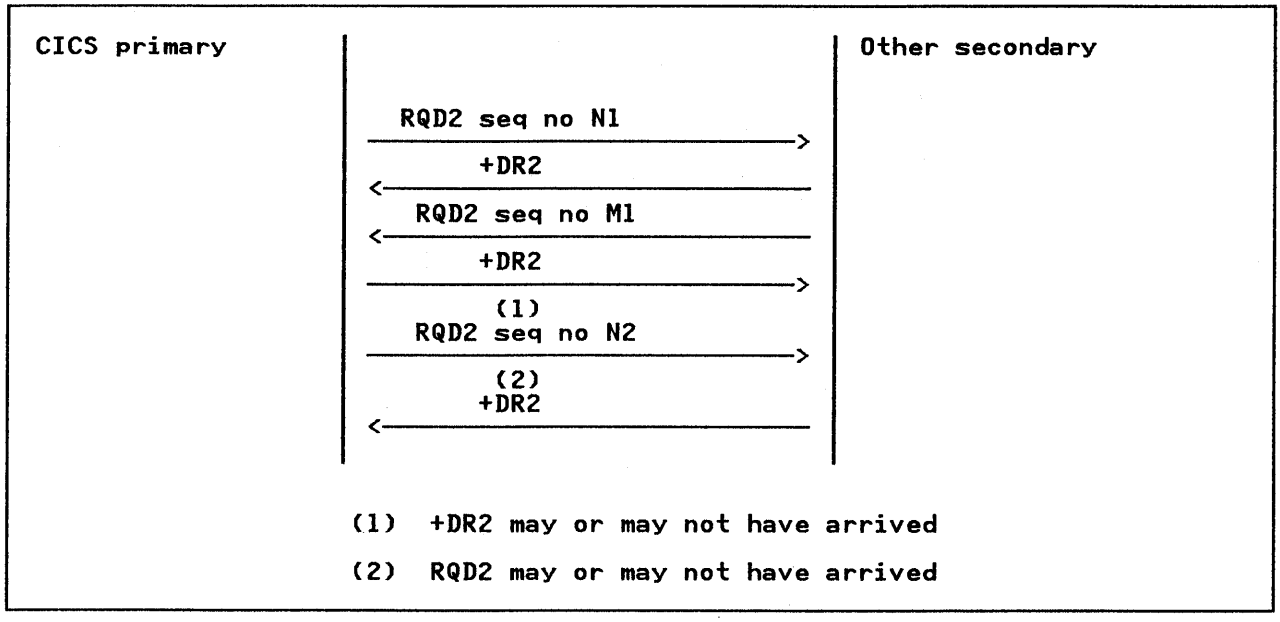


Figure 144. STSN Flows before Failure (CICS Primary)

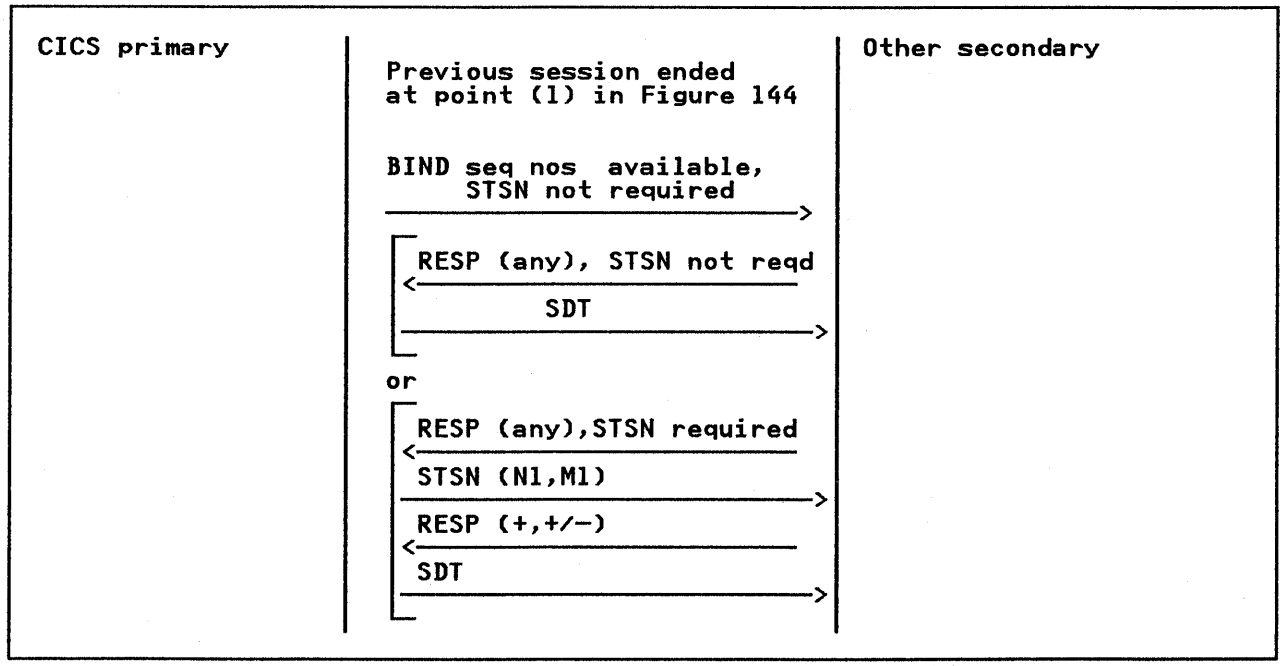


Figure 145 (Part 1 of 4). STSN Flows for Session Restart (CICS Primary)

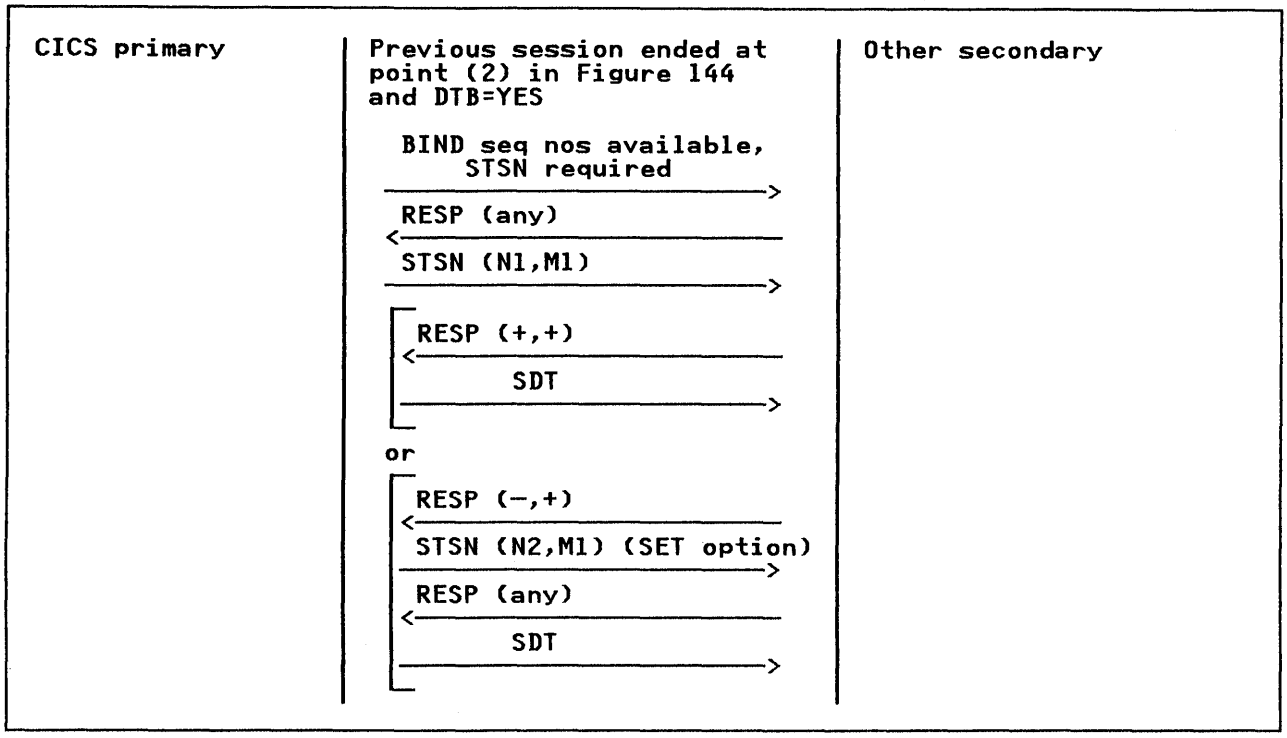


Figure 145 (Part 2 of 4). STSN Flows for Session Restart (CICS Primary)

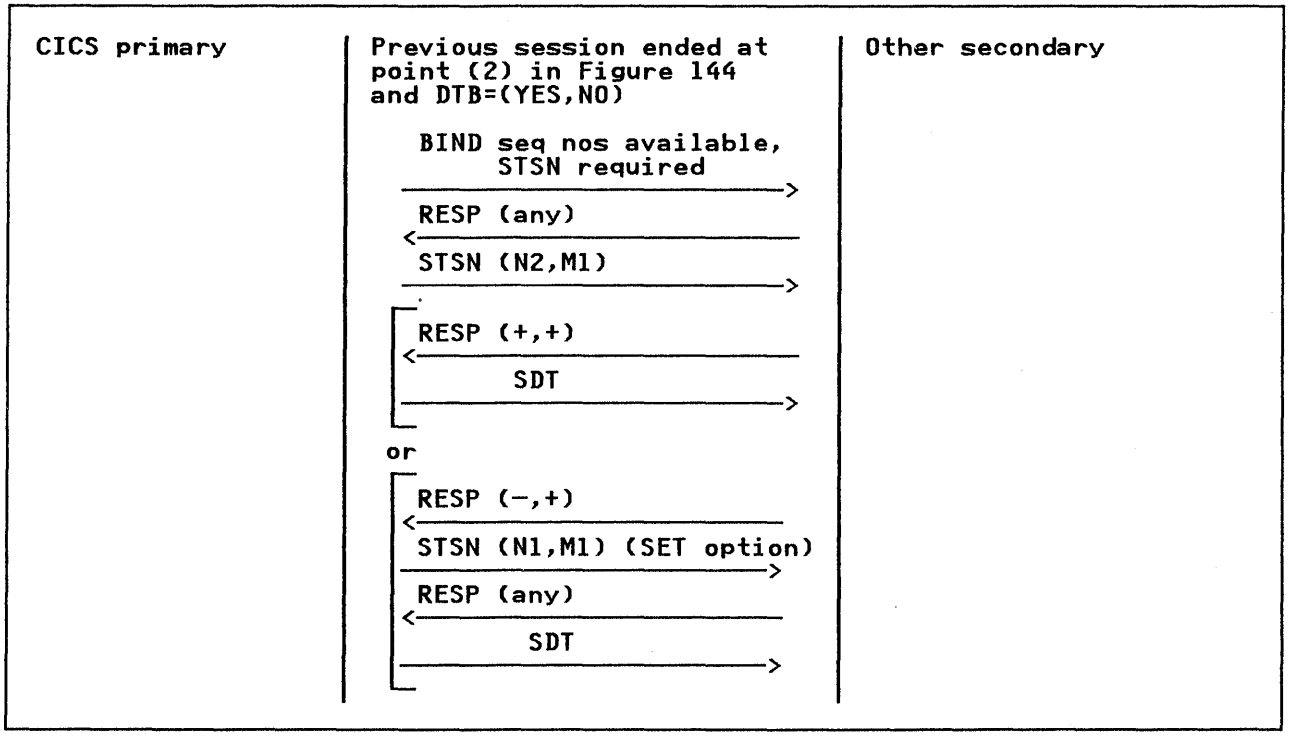


Figure 145 (Part 3 of 4). STSN Flows for Session Restart (CICS Primary)

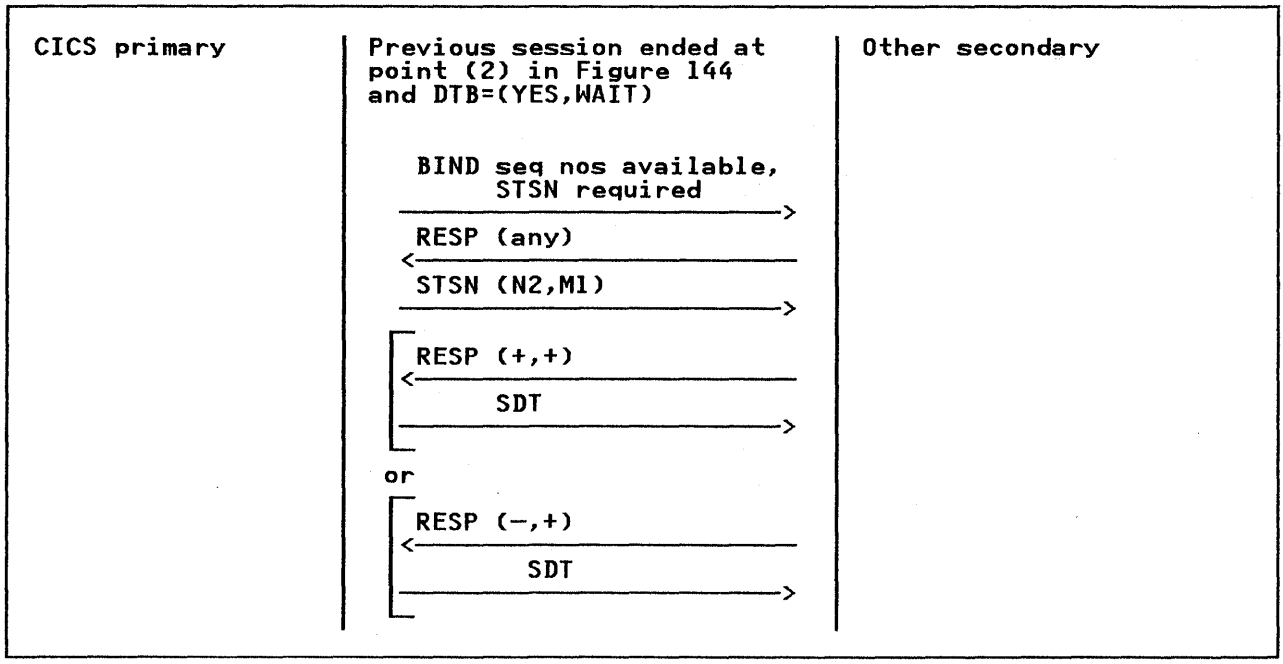


Figure 145 (Part 4 of 4). STSN Flows for Session Restart (CICS Primary)

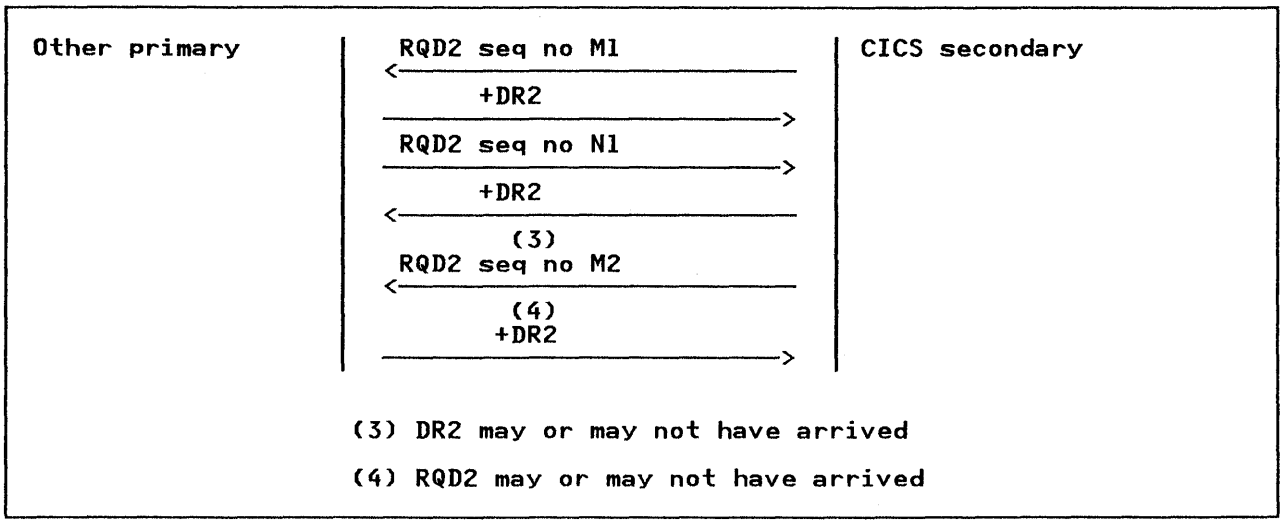


Figure 146. STSN Flows before First Type of Session Failure (CICS Secondary)

Figure 147 on page 313 shows flows at session restart for:

1. CICS as secondary.
2. CICS has memory of the previous use of the session, that is the session ended normally or abnormally during this CICS run, or there has been an intervening emergency restart.
3. The previous session ended at points (3) or (4) in Figure 146.

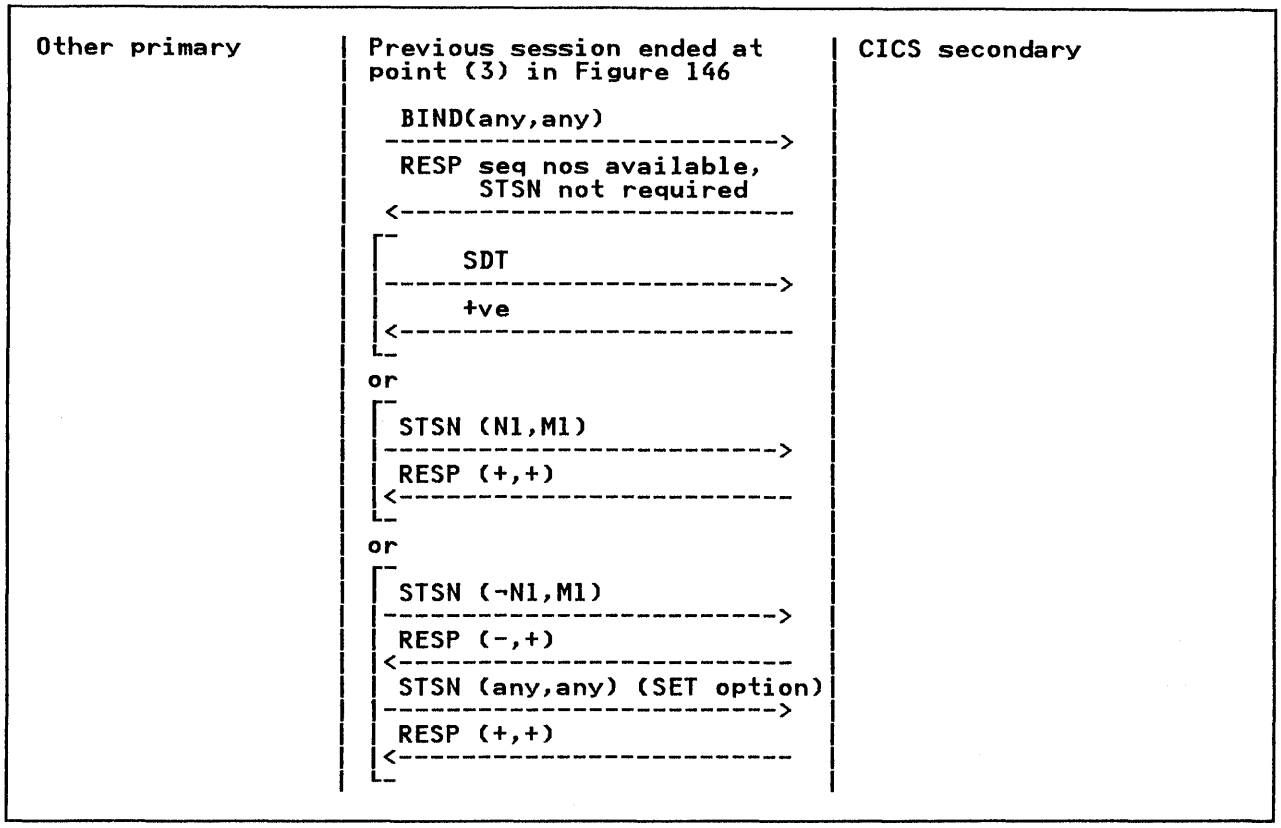


Figure 147 (Part 1 of 4). STSN Flows for First Type of Session Restart (CICS Secondary)

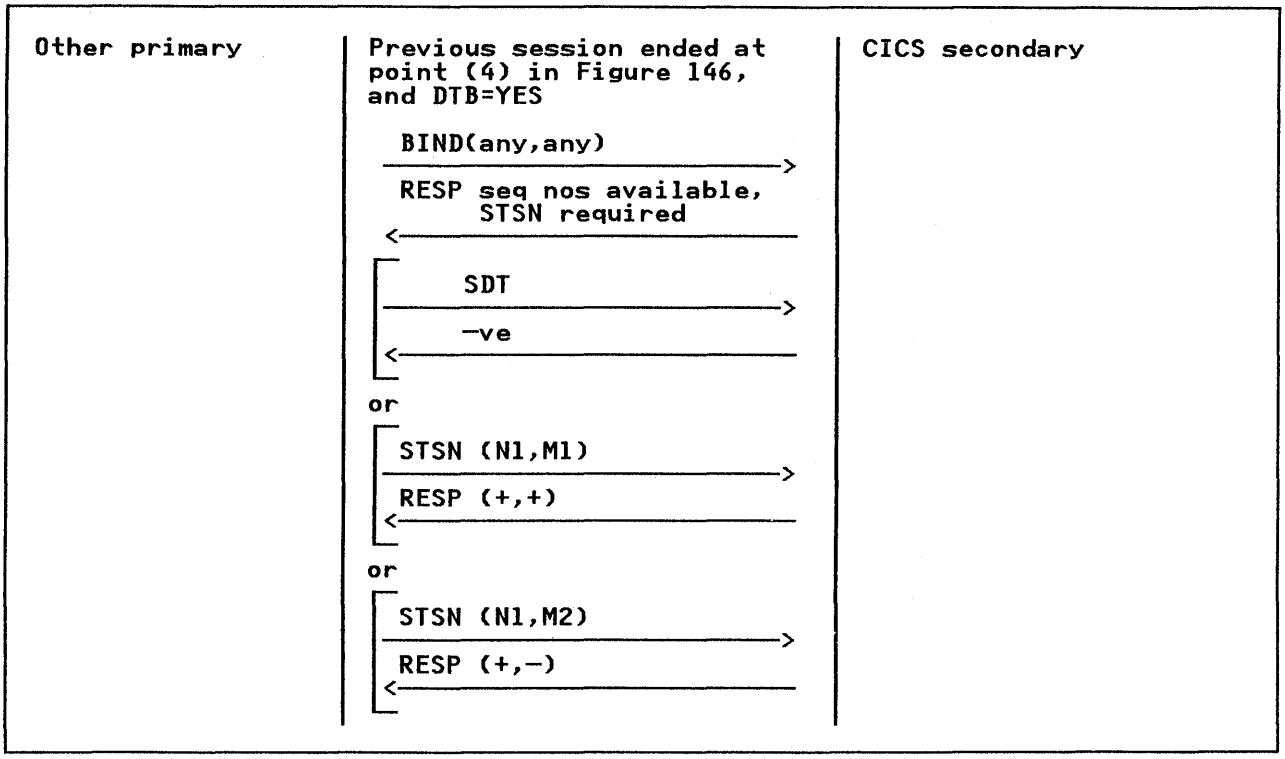


Figure 147 (Part 2 of 4). STSN Flows for First Type of Session Restart (CICS Secondary)

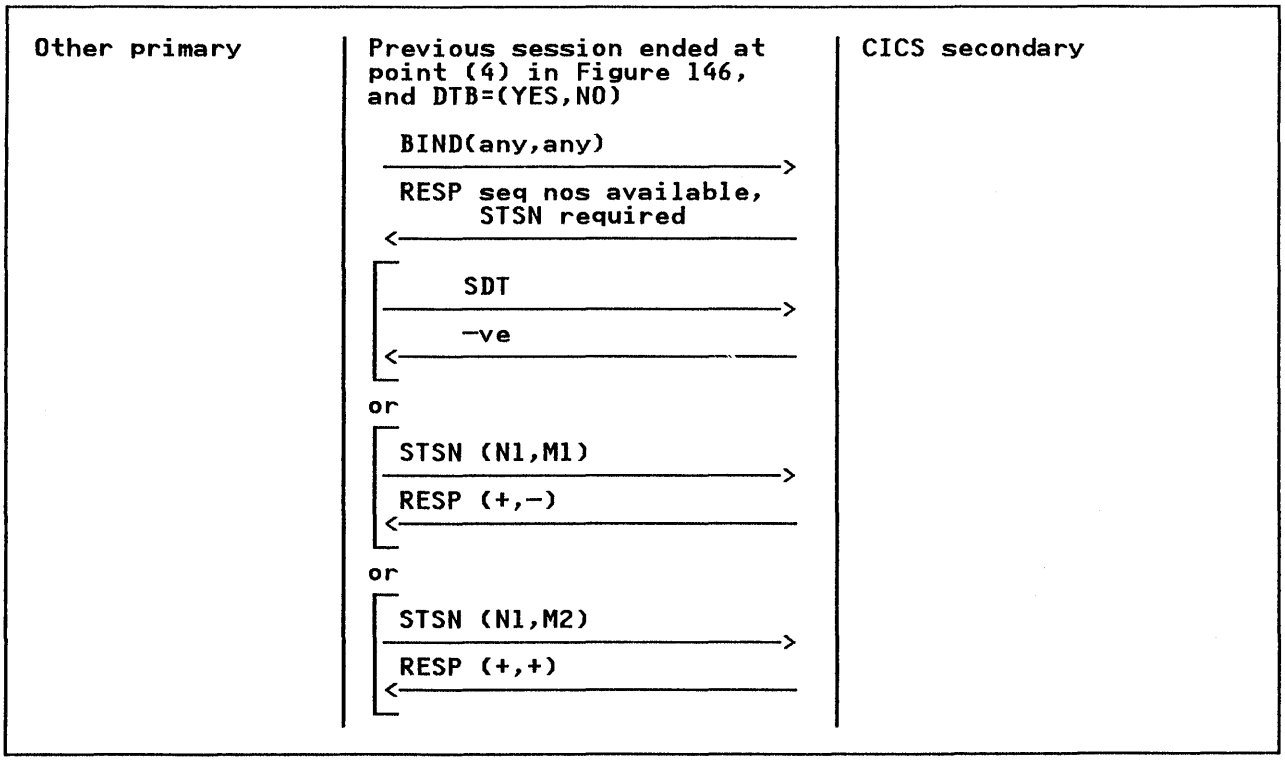


Figure 147 (Part 3 of 4). STSN Flows for First Type of Session Restart (CICS Secondary)

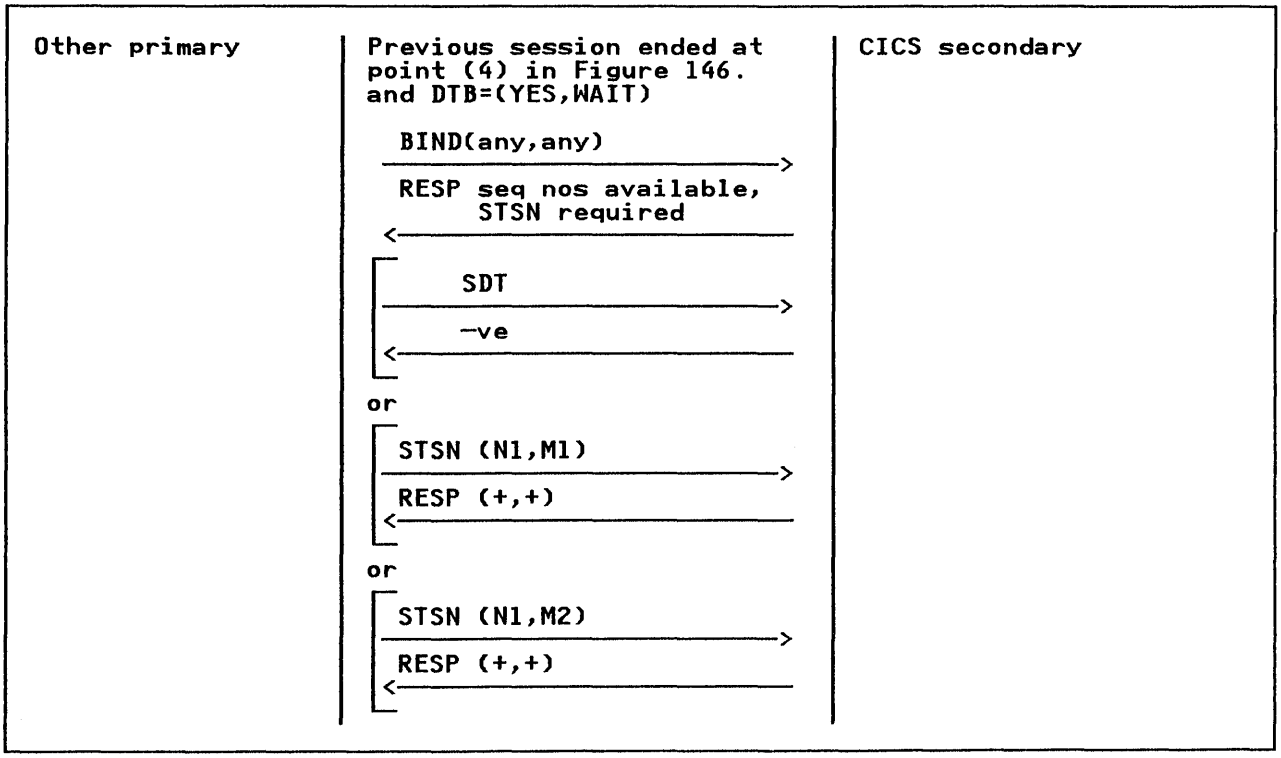


Figure 147 (Part 4 of 4). STSN Flows for First Type of Session Restart (CICS Secondary)

Figure 149 shows flows at session restart for:

1. CICS as secondary.
2. CICS has memory of the previous use of the session, that is the session ended normally or abnormally during this CICS run, or there has been an intervening emergency restart.
3. The previous session ended at point (5) in Figure 148.

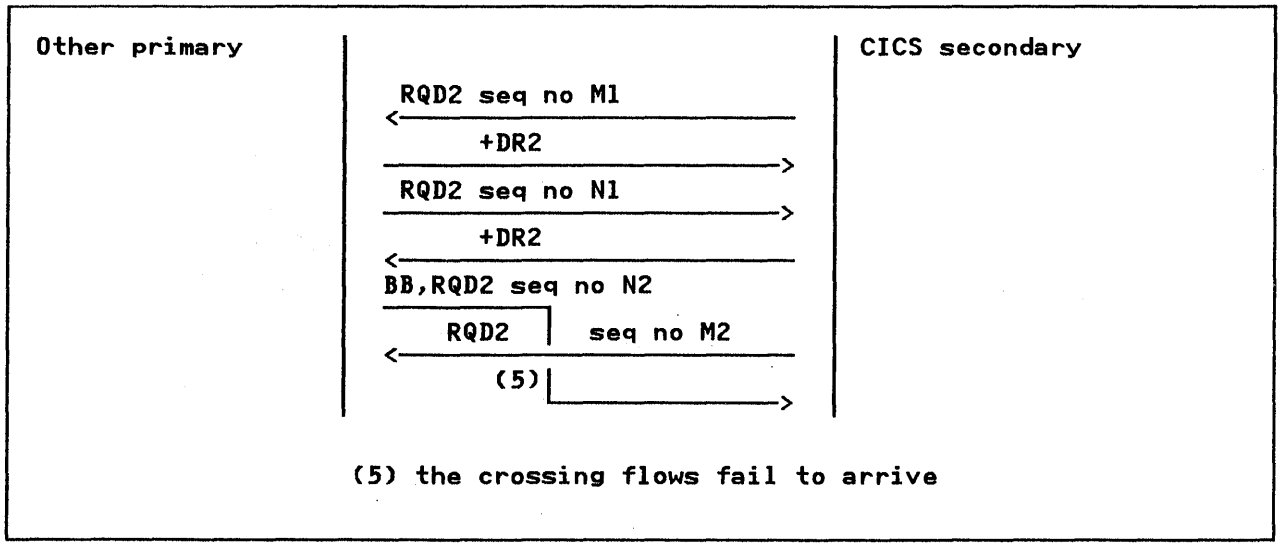


Figure 148. STSN Flows before Second Type of Session Failure (CICS Secondary)

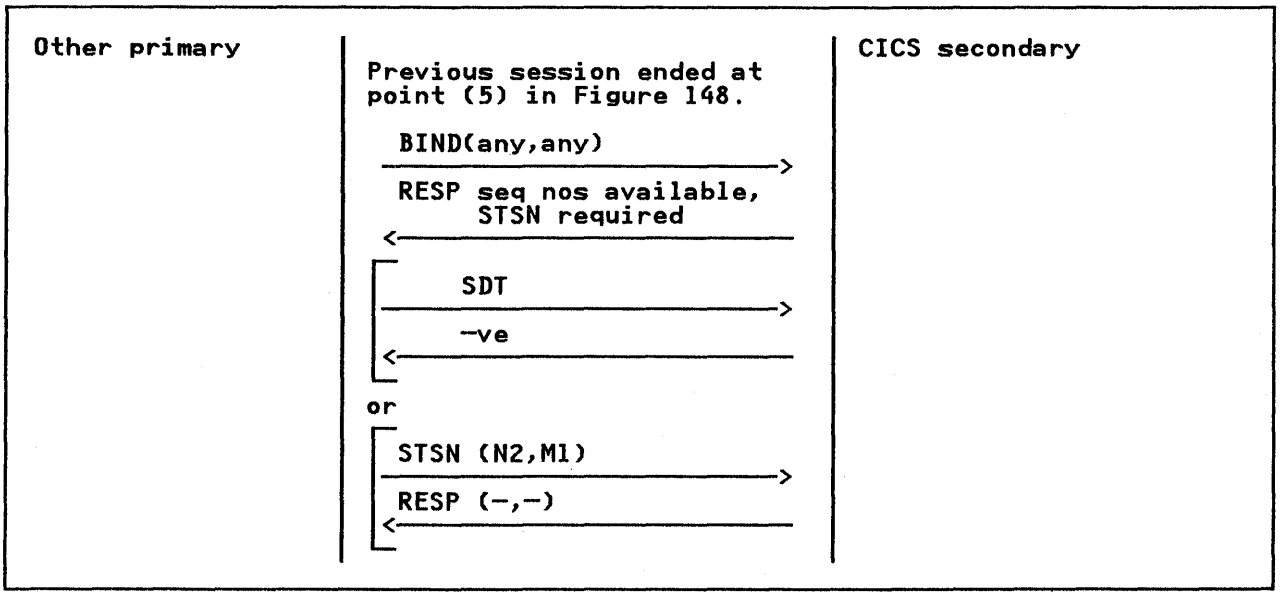


Figure 149. STSN Flows for Second Type of Session Restart (CICS Secondary)

Session Shutdown

CICS supports a two-phase stop bracket initiation (SBI), bracket initiation stopped (BIS) exchange protocol. SBI and BIS are SNA commands. On sending or receiving SBI, CICS enters a session quiesce state. This means that CICS is selective about the brackets that it initiates. The other subsystem should suppose that there is no detectable difference from normal session execution between SBI send/receive and BIS send.

CICS, after sending or receiving SBI (or both), permits the other subsystem to start any brackets that it wishes to. The assumption is that, since SBI was seen by the other subsystem, BIS will be forthcoming eventually.

Figure 150 and Figure 151 show flows at session shutdown.

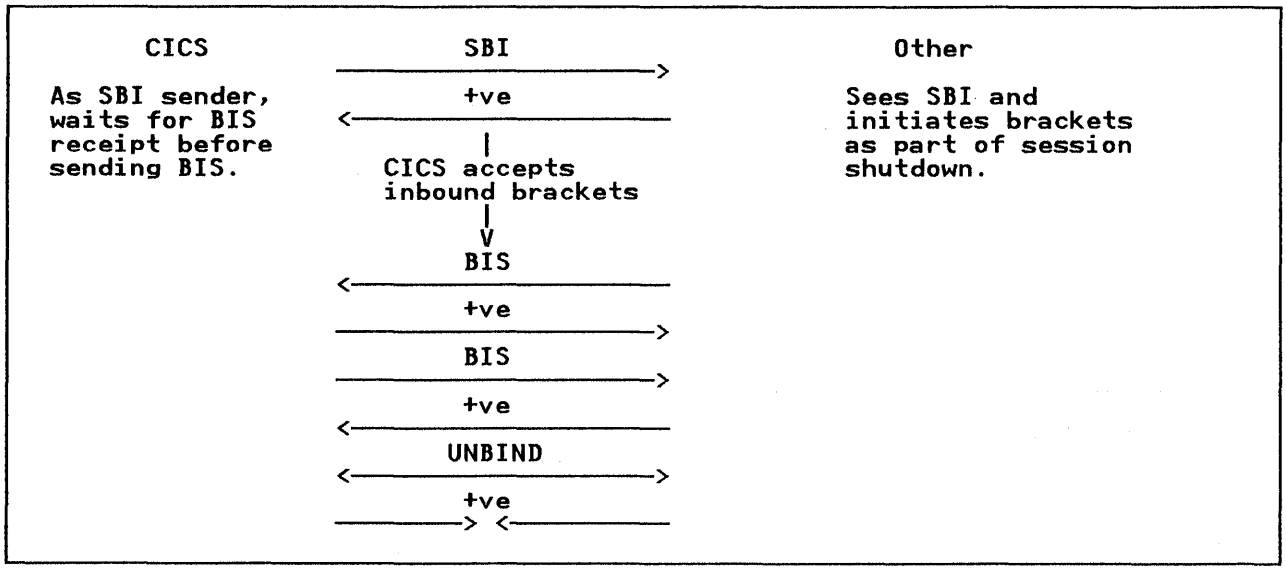


Figure 150. Normal Shutdown, CICS SBI Sender

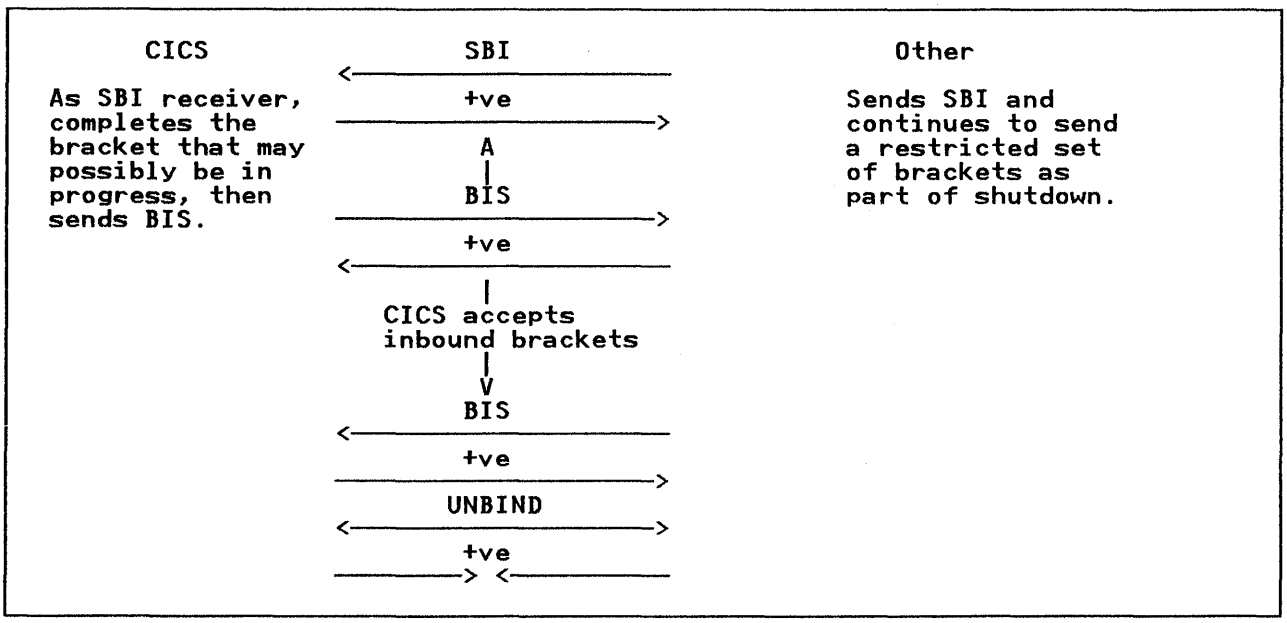


Figure 151. Normal Shutdown, CICS SBI Receiver

DFC Indicators for Error Recovery

CICS uses negative response (0846) to enter the receiver (ERP) state. This code indicates that a message describing the failure is forthcoming.

A half-session does not always wish to enter the ERP state solely because of session events. The desire to enter the ERP state is often brought about by a process that is executing within the half-session failing for some (often user-detected) reason. There can be a multiplicity of reasons for such a failure. In general there may be no obvious correlation between the error and the activity on the session. Indeed, the process may fail for reasons entirely unrelated to session activity.

Note: A session error of any kind, when transmitted to the (so far) nonfailing half-session, must be communicated to the nonfailing process. This action may or may not provoke the nonfailing process into failure – the effect of this is that there may well be a tendency for some systems to view an entry into the ERP state as a symmetric activity.

Both sides may indicate a desire to enter the receiver ERP state in an essentially symmetrical way.

The (0846) negative response sender is, because of half-duplex flip-flop protocol, bound to end up in the ERP send state, or to be RESET by moving to the BETB state. In order to move into the ERP send state, a purging state must be passed through. This terminates either directly (if the response was to an RQD or CD chain), or at the first synchronizing flow received (RQD or CD), or BETB. CICS negatively responds (0867) to the synchronizing event, unless it is a command.

Since CICS does not transmit an ERP message BB,EB, there is the possibility of (0846) losing information. CICS tries to avoid this situation by responding with the more informative code if the BETB state is a necessary outcome of the current bracket state.

Purging and Resynchronization

The ERP pending state to ERP send state transition is now described. At the sender's half-session, this is resynchronization logic; at the receiver's half-session, this is purging logic. CICS makes the assumption that these activities are half-session supervisor activities.

For the sender's half-session:

1. The session may already be synchronized. This is true if CD has been received and no outbound chains have been sent. It can also occur because a response was received to the last outbound chain. If the last outbound chain is RQD*, then synchronization is achieved by waiting for the response.

2. If the last outbound chain was sent requesting an exception response and a response to it has not yet arrived, then a CHASE command is sent as a generated synchronizing event.

3. If the current outbound chain is incomplete, then a CANCEL command is sent as a generated synchronizing event.

4. The resynchronization just described may cause the half-session to receive a –(0846) response. In that case, the half-session enters the receive state. This is the receiver ERP rule that the first –(0846) is always the one to determine the ERP send/receive state.

For the receiver's half-session:

1. The receiver sends a negative response –(0846). If there happens to be a request unit available, then this is used. This is not the case if the last flow was an outbound CD chain, or if the last inbound chain was RQD and the response has already been sent. In this case, the half-session supervisor waits for an inbound request unit after issuing a SIGNAL command.

2. Once the negative response has been sent, the RECEIVE half-session purges to a synchronizing event. This is null if already synchronized. The session is already synchronized if the –(0846) was to an RQD chain.

3. A synchronizing event is any one of:

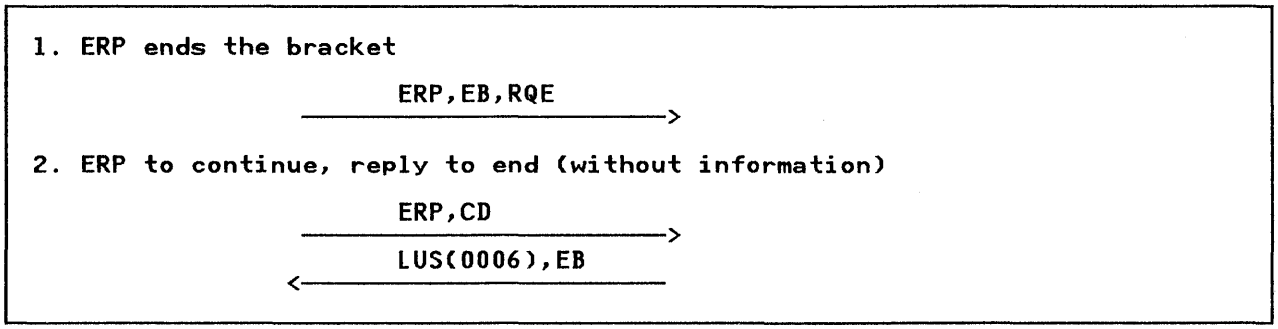
   a. An RQ*, CD chain
   b. An RQD, FMD chain
   c. An LUS command sent RQD
   d. A non-LUS command sent RQD.

   In each case the synchronization is completed by the following responses:

   a. –(0867) or –(0846)
   b. –(0867) or –(0846)
   c. –(0867) or –(0846)
   d. +ve

4. The receiver may receive a –(0846) before purging has completed. This is the case when the last thing sent was an RQ*,CD chain and nothing has yet been received. If a –(0846) response is received to this chain, the half-session enters the receive state.

ERP Examples



Receive State Errors

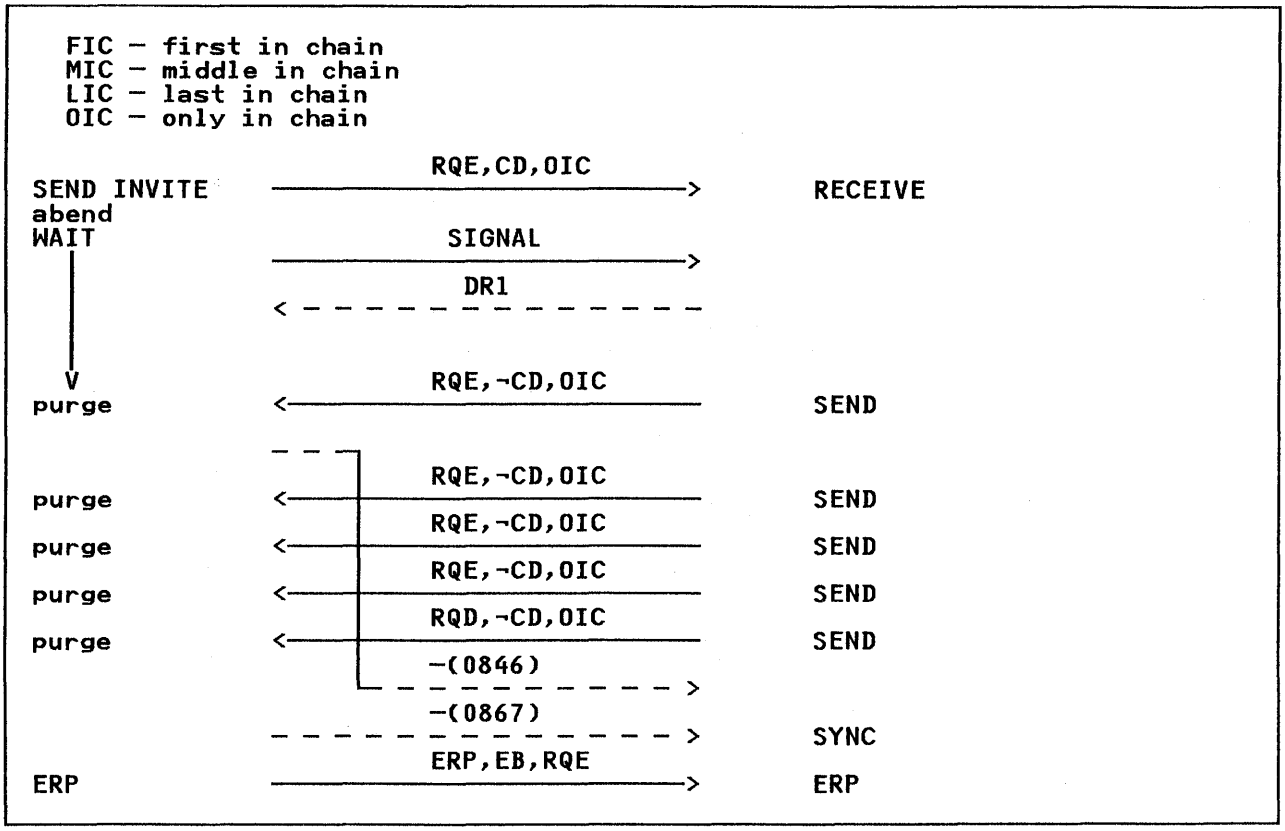


Figure 152. Receive State Errors, CD Sent, Sync Event = RQD

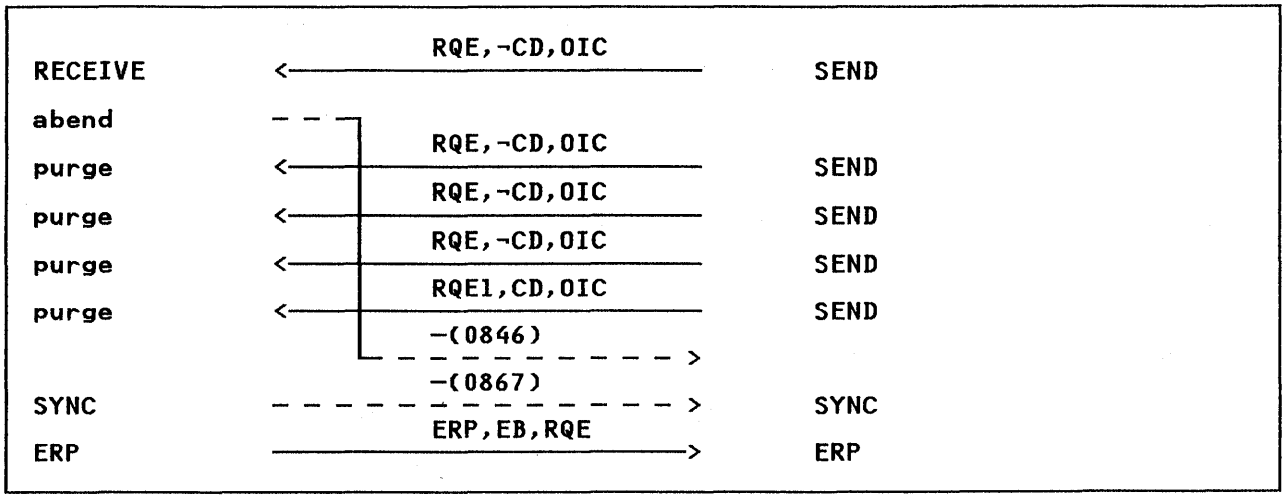


Figure 153. Receive State Errors, ¬CD Received, Sync Event = CD

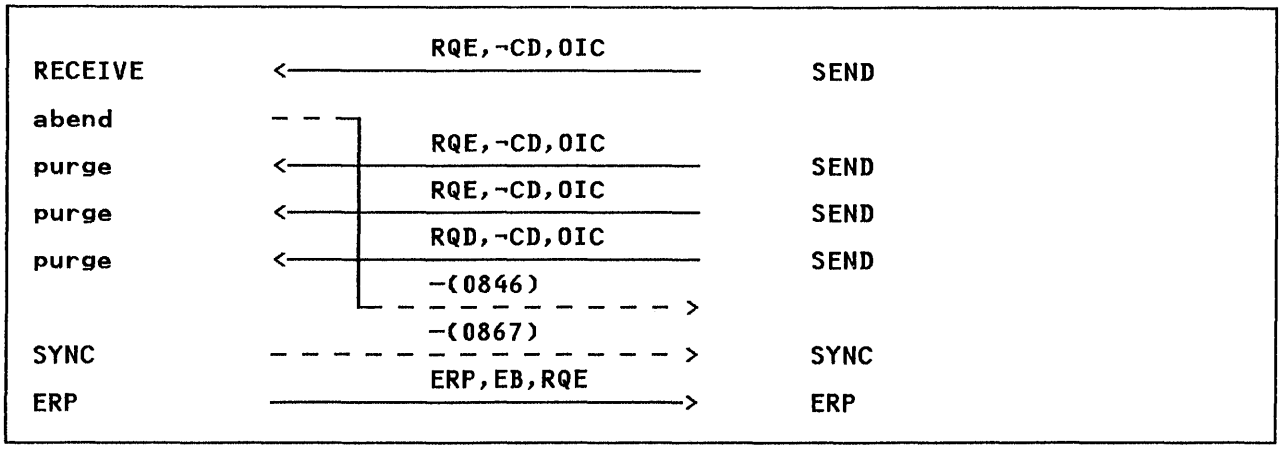


Figure 154. Receive State Errors, ¬CD Received, Sync Event = RQD

Send State Errors

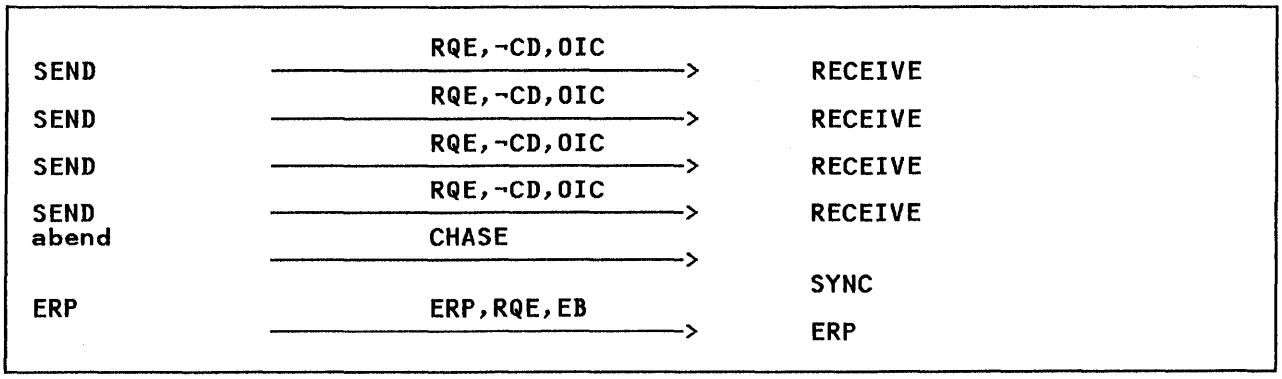


Figure 155. Send State Errors, BETC, Exception Request Sent

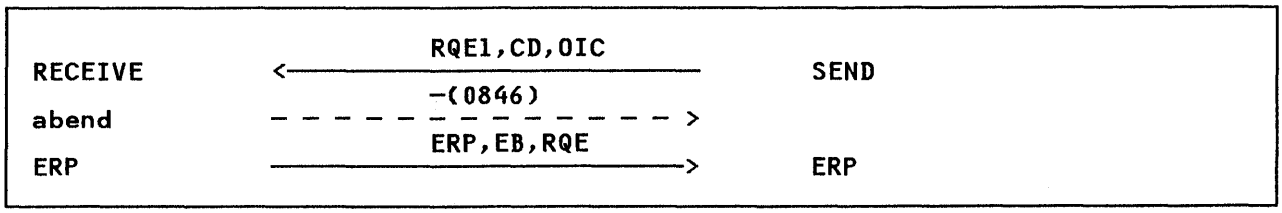


Figure 156. Send State Errors, CD Received, RQE1

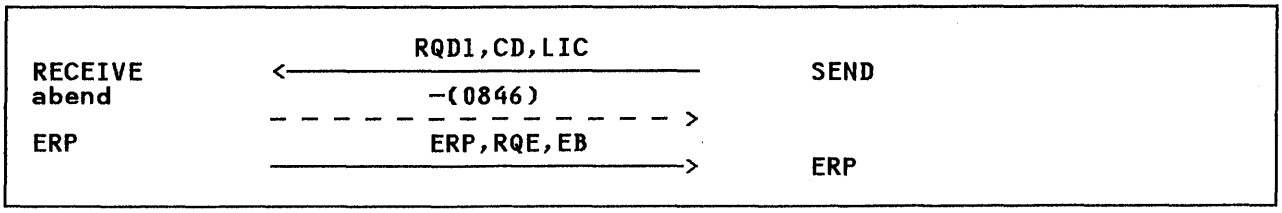


Figure 157. Send State Errors, CD Received, RQD1, Response Not Sent

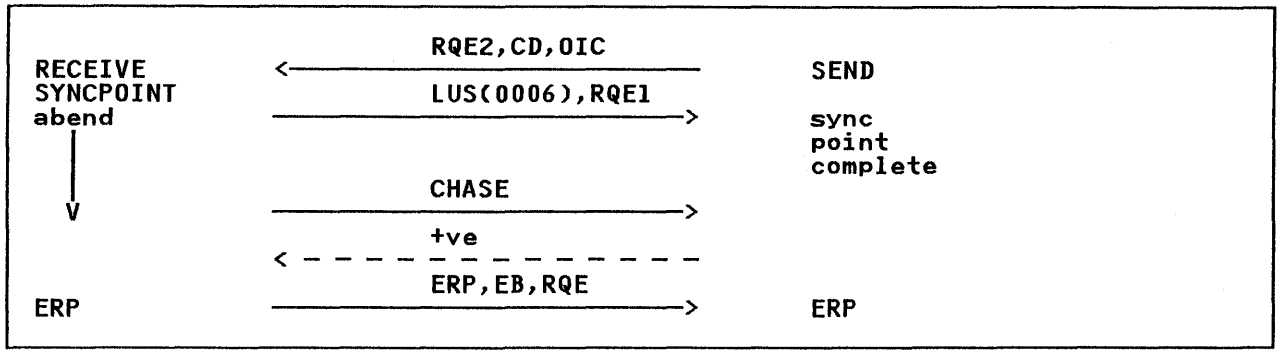


Figure 158. CD Received, RQE2

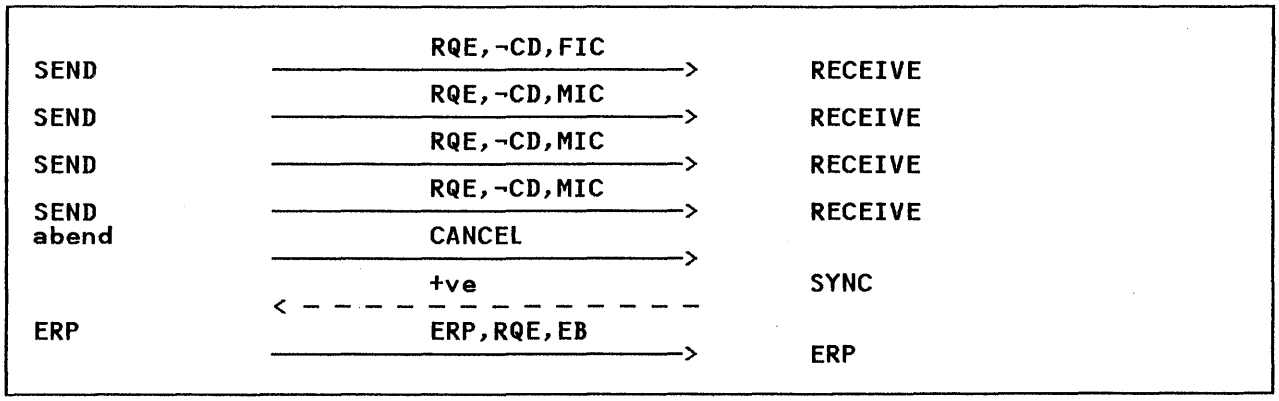


Figure 159. Send State Errors, In-Chain

Errors in Each Half-Session

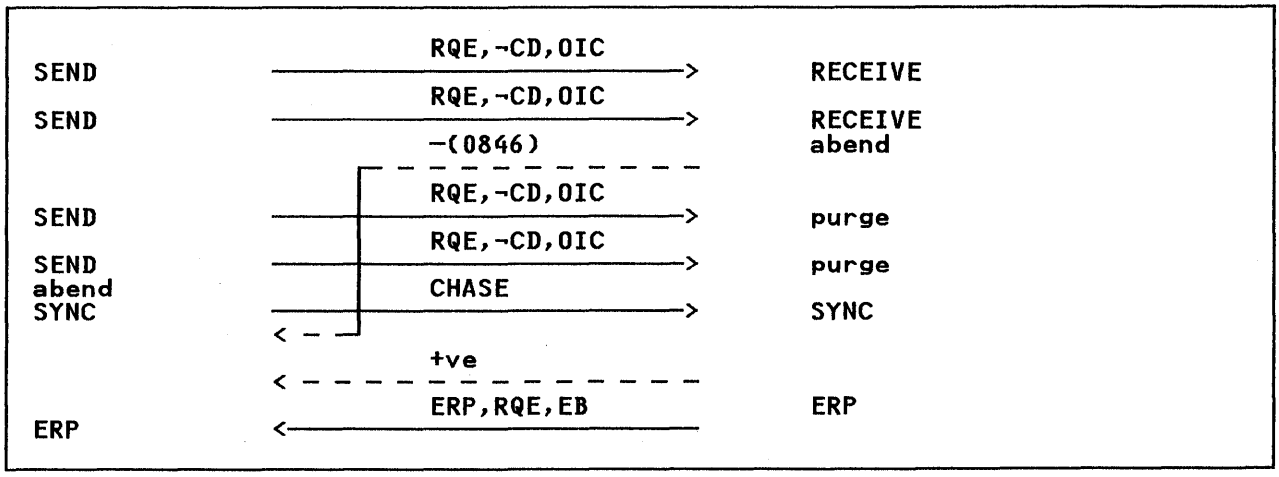


Figure 160. Half-Session Errors, BETC, Negative Response in Pipe

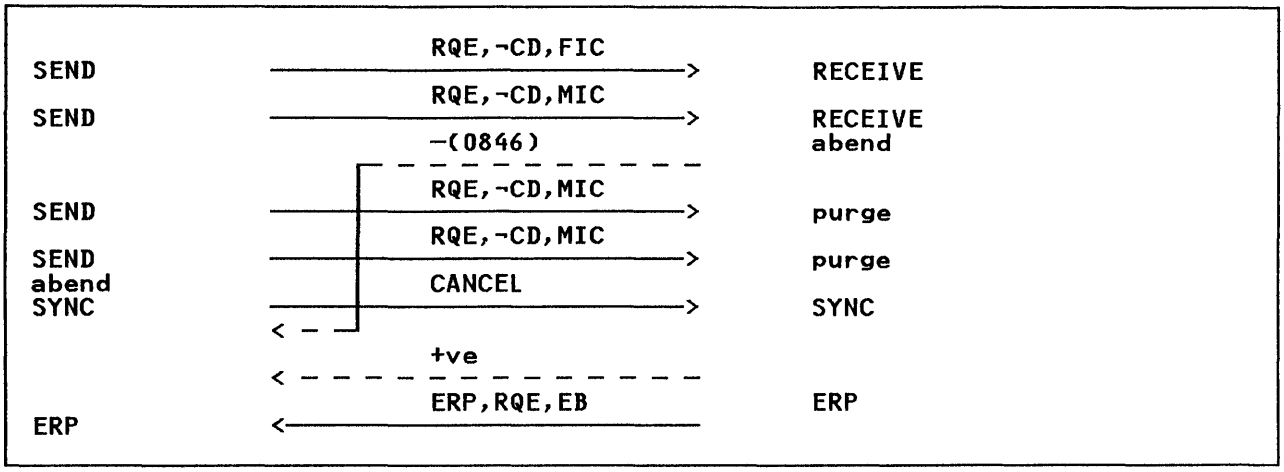


Figure 161. Half-Session Errors, In-Chain, Negative Response in Pipe

Chapter 2.10. Intercommunication Using LU6.2

LU6.2 is designed to have a basic level of function, which all licensed programs must provide, plus a number of optional facilities which can be implemented if desired. The basic level is called the BASE and the optional facilities are called OPTION SETS above the BASE.

A description of some of the major facets of the CICS implementation follows, to provide a framework for the description:

- Generalized data stream (GDS)

  In a generalized data stream (GDS), the data is preceded by LLID, where LL is the overall length of each GDS data variable and ID describes the data that follows.

- Conversation types

  Two types of conversation across an LU6.2 session are defined:

  – Plain conversations, also called unmapped conversations.

    In this conversation type, the user needs to understand GDS formats, and to code data streams containing LLIDs.

  – Mapped conversations.

    In this conversation type, the user is passed only the data, that is, without the LLID. The user has no knowledge of the IDs being used and is not responsible for calculating LLs.

    Note: This mapping must not be confused with BMS mapping.

  A command-level language interface, such as the CICS ISC application program interface (API), uses mapped conversations.

- Layered design

- Use of LU6.2 by LU6.1 programs

  Programs written for LU6.1 use mapped conversations. The data stream built by the application program is encapsulated by CICS into GDS entities. If FMHs are used between application programs, they are encapsulated into separate entities using a special GDS ID. Thus, the data stream appears to be a normal GDS data stream. At the remote system, the GDS IDs are removed by CICS, prior to presenting the data to the remote program in the form it is expecting.

- Synchronization levels

  LU6.2 provides a means for application programs to determine the level of synchronization required and to control when such synchronization is performed. When one transaction starts a conversation, it specifies the level of synchronization required. The levels are:

  – NONE

    In this case no synchronization takes place between transactions.

- COMMIT ONLY

  In this case, CICS transactions manage synchronization themselves, by issuing SEND CONFIRM/ISSUE CONFIRMATION commands. These commands are exchanged between two user programs, and the CICS sync point mechanisms are not involved in any way.

- ALL

  This is equivalent to normal CICS SYNCPOINTing, including ROLLBACK if needed. When an EXEC CICS SYNCPOINT command is issued, CICS ensures synchronization of all protected resources.

- Communicating errors between application programs

  It is possible for a transaction to issue a command to indicate to the other transaction that something has gone wrong. The second application can then take appropriate action. In CICS, this is the ISSUE ERROR command.

- LU services manager

  This is required for any LU6.2 system that supports parallel sessions or that supports a sync point level of ALL, and is needed, for instance, to change the number of sessions between systems.

CICS Implementation

Plain Conversations

Plain conversations are executed via the GDS API. GDS commands are handled by DFHEIP which passes control to DFHEGL. The control of flow is shown in Figure 162. Module descriptions are given under "Modules" on page 328.

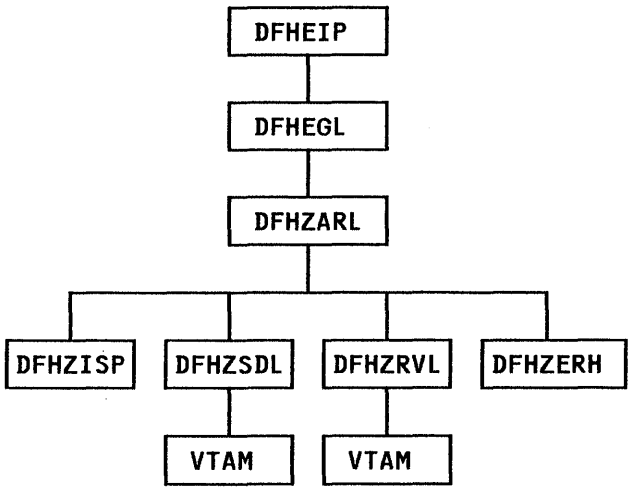


Figure 162. Terminal Management Flow for LU6.2 (Plain Conversations)

- DFHEIP routes all GDS commands to DFHEGL, which invokes DFHZARL.

- DFHZARL is always invoked via the DFHLUC macro. The DFHLUCDS DSECT maps a data area that is set up to pass information to and return information from DFHZARL. DFHZARL manages data in buffers, not TIOAs. SEND commands cause data to be assembled by DFHZARL into a buffer until a WAIT, or other event, causes the buffer to be transmitted.

  - DFHZARL invokes DFHZSDL and DFHZRVL to exchange data with VTAM, by placing requests on the activate chain. However, for optimization, DFHZARL can invoke DFHZSDL directly.

  - DFHZERH is called by DFHZARL, either when it is required to transmit error information, or when error information has been received.

  - DFHZISP is called by DFHZARL to perform ALLOCATE requests.

Mapped Conversations

Mapped conversations use the normal CICS API. Application programs and function shipping requests written for LU6.1 operate using mapped conversations when transferred to LU6.2.

The control of flow is shown in Figure 163. Module descriptions are given under "Modules" on page 328.

- DFHETC passes control to DFHETL if it detects that an LU6.2 session is in use. (ALLOCATE commands are passed to DFHZISP since DFHETC cannot check the session type until the session is allocated.)
- DFHETL routes the commands to the appropriate module:
  - FREE commands to DFHZISP.
  - SEND, WAIT, CONVERSE and some RECEIVE commands to DFHZARQ. RECEIVE commands are passed to DFHZARQ if input journaling is in effect. Otherwise, the call is routed to DFHZARL directly.
  - Commands that are only valid for LU6.2 (for example, CONNECT PROCESS and ISSUE ERROR) are passed directly to DFHZARL.
- DFHETL may invoke DFHZARM to provide service functions.
- DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if it finds that an LU6.2 session is in use. This applies to the SEND, RECEIVE and SIGNAL commands. The same applies to DFHZISP for the FREE command, and DFHSPP for a FLUSH command.
- DFHZARM performs the translation of the data stream to and from a format suitable for invoking DFHZARL. In particular:
  - An LU6.2 attach FMH may have to be requested.
  - Data must be passed in GDS format (structured fields preceded by an LLID).

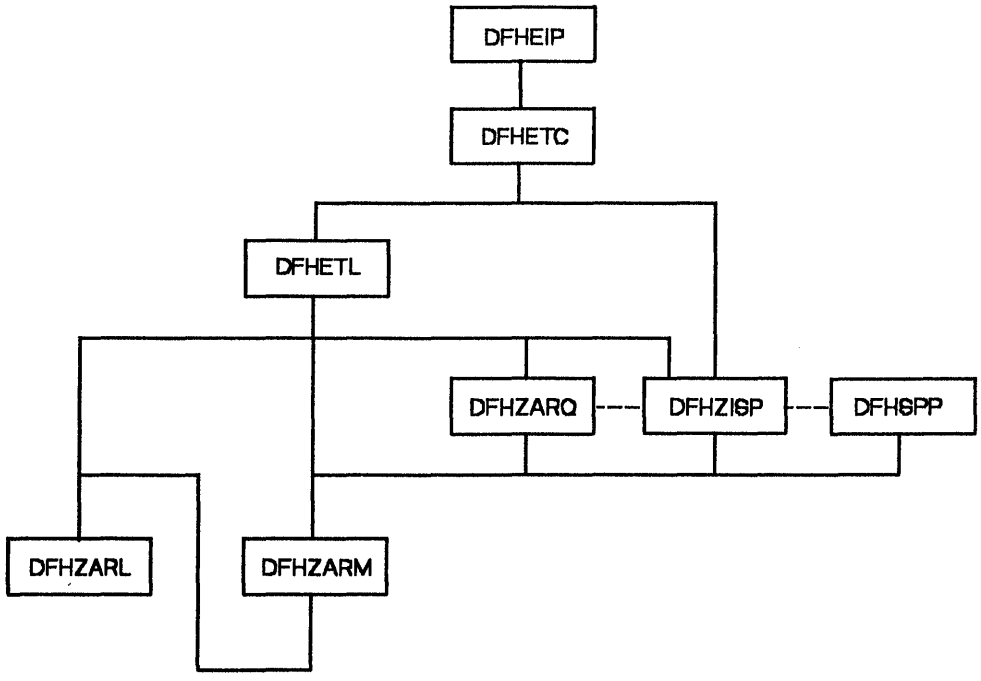


Figure 163. Terminal Management Flow for LU6.2 (Mapped Conversations)

– Chaining differences are handled. In LU6.1, one SEND command is equivalent to one chain. In LU6.2, a chain continues until the direction of transmission is reversed, for example, when a RECEIVE command is issued after a SEND command.

The result is that data is passed across the API in the same way for both LU6.1 and LU6.2. The differences in what is transmitted are handled by CICS.

LU6.2 Session States

In order to remember the state of a session at any time, four modules maintain the following states:

Module	State	Macro
DFHZUSR	Conversation – for example, SEND, RECEIVE	DFHZUSRM
DFHZBKT	SNA bracket state	DFHZBSM
DFHZCNT	Contention state	DFHZCNM
DFHZCHS	Chaining state	DFHZCHM

These modules are invoked via the macros shown in the last column. Any query or change to the states is performed using these macros.

The modules are usually referred to as **state machines**. When a module, such as DFHZARL, wishes to check that the session is in a suitable state to perform a given operation, it asks the appropriate state machine to perform the check, using the relevant macro. If the operation subsequently causes a change in the state, the state machine is again invoked to record the new state. The LU6.2 states for each session are stored in the TCTTE for that session.

In the case of LU6.1, another set of bits in the TCTTE is manipulated directly by the modules which handle state changes. DFHZARM and DFHETL manipulate these bits for mapped conversations, on the basis of information supplied to or returned from DFHZARL. This permits modules such as DFHZARQ to operate independently of whether an LU6.1 or LU6.2 session is in use, since they only need to work with the LU6.1 state indicators.

The most complex of these modules is DFHZUSR – the user state machine. It is controlled by DFHZARL and controls user requests. The states are:

- NOT_ALLOCATED
- ALLOCATE_IN_PROGRESS
- ALLOCATED_SEND
- ALLOCATED_RECEIVE_PENDING
- ALLOCATED_RECEIVE
- FREE_PENDING_SEND
- FREE_REQUIRED
- IN_SYNCPT_SENDER_ONE_PHASE
- IN_SYNCPT_RCVER_ONE_PHASE
- IN_SYNCPT_SENDER_TWO_PHASE
- IN_SYNCPT_RCVER_TWO_PHASE
- IN_SYNCPT_BACKOUT_SENDER
- IN_SYNCPT_BACKOUT_RECEIVER
- ALLOCATED_CONFIRM_SENDER
- ALLOCATED_CONFIRM_RECEIVER

Note: This machine gives the user's view of the state of the conversation which, because of the deferred I/O, will not necessarily be the same as the state of the SNA session.

Modules

The modules that handle LU6.2 are as follows:

- DFHEGL
- DFHETL

- DFHZCC which contains:
  - DFHZARL
  - DFHZARM
  - DFHZRVL
  - DFHZRLX
  - DFHZSDL
  - DFHZSLX
  - DFHZBKT
  - DFHZCHS
  - DFHZCNT
  - DFHZUSR
- DFHZCW which contains:
  - DFHZERH
  - DFHZLUS
- DFHLUP

DFHEGL

DFHEGL performs the processing of GDS commands. It is a normal EXEC stub and receives control directly from DFHEIP. The TCTTE for the session is located and checked for validity. All GDS conversation related commands are mapped into a DFHLUC macro call and routed directly to DFHZARL. There is no mapping or unmapping of data, state indicators are not maintained, and there are no FMHs to process. Some validation of application provided parameters is performed and errors are reflected back to the application.

DFHETL

DFHEIP routes all terminal control requests to DFHETC. DFHETC routes all requests relating to an LU6.2 session to DFHETL, except ALLOCATE which is routed to DFHZISP, and BUILD ATTACH, EXTRACT, and POINT, which are handled by DFHETC itself. DFHETL performs the following actions:

1. Maps an application request into a form suitable for the DFHZCP/DFHZCC application request modules.
2. Detects errors and returns error codes to the application.
3. Unmaps data.
4. Maintains state indicators.

For the commands ISSUE CONFIRMATION, CONNECT PROCESS, EXTRACT PROCESS, ISSUE ERROR, ISSUE ABEND, and ISSUE SIGNAL, DFHETL:

1. Maps application requests into DFHLUC macro calls.
2. Updates state indicators in the TCTTE (for example, the TCTTE indicator that states that a CONNECT PROCESS has been issued).
3. Flushes deferred data for ISSUE ERROR and ISSUE ABEND commands using a DFHLUCM call.

For SEND and CONVERSE commands, DFHETL:

1. Calls DFHZARM to flush deferred data.
2. Obtains storage for the processing of outbound application data.
3. Creates attach FMHs, if appropriate.
4. Calls DFHZARQ to transmit data.
5. Calls DFHZARL if the CONFIRM option is specified.

Reasons for calling ZARQ are:

- To avoid duplication of existing code.
- So that DFHZCP performs journaling of outbound data.
- To perform an implicit CONNECT PROCESS if SEND or CONVERSE is the next session-related command after ALLOCATE.
- To enable the piggyback of CD or EB.

For RECEIVE commands, DFHETL:

1. Obtains storage for the processing of inbound data.
2. Calls DFHZARQ if inbound journaling is required, otherwise DFHZARL is called to receive inbound data.
3. Extracts inbound FMHs, as appropriate.
4. Unmaps inbound data.
5. Validates LLs and rejects them if invalid.
6. Manages the passing back of data to the application.
7. If the application issues a RECEIVE NOTRUNCATE request in order to receive only part of the chain, retains the residual data for subsequent RECEIVE(s). DFHETL receives a complete chain of data at a time from DFHZARL or DFHZARQ.

For WAIT commands, DFHETL:

1. Calls DFHZARQ.

For FREE commands, DFHETL:

1. Checks that the terminal is in the correct state to be freed.
2. Frees the storage used to hold RECEIVE data and the ETCB.
3. Calls DFHZISP to free the session.

Reasons for invoking DFHZISP are:

- To be symmetric with DFHETC which calls DFHZISP for the ALLOCATE.
- To perform an implicit SEND of EB, if required.

DFHZARL

DFHZARL is invoked using the DFHLUC macro. This section describes the processing for the principal functions.

INITIAL_CALL Function

This function is requested by DFHZSUP. DFHZARL acquires a send and receive buffer. The length is always a multiple of the RU size, and is normally less than 4K bytes (unless the RU size exceeds 4K bytes). If the transaction is being started as a result of an attach request received from a remote system, DFHZARL transfers any data received with the attach header from the TIOA into the receive buffer.

Allocate Function

DFHZARL performs the following actions:

1. Copies the input parameters to LIFO storage.
2. Calls DFHZISP to allocate a TCTTE.
3. Addresses the TCTTE allocated.
4. Sets the user state machine (DFHZUSRM).

   (request = ALLOCATE_RESOURCE)
5. Returns to the caller.

SEND Function

DFHZARL performs the following actions:

1. Checks the user state machine.
2. Checks the LL count and maintains a record of the outstanding LL count.

3. If the command is SEND LAST, INVITE, or CONFIRM, and the outstanding LL count is not zero, issues an error message.

4. Sets the user state machine.

The caller must specify WAIT in the request to ensure that the data is sent synchronously. SEND CONFIRM has an implicit WAIT and control is not returned until a response has been received, when the state machine is set.

For a SEND request with WAIT, DFHZARL then:

1. Sets the user state machine.

   (request = WAIT)

2. Invokes DFHZSDL for transmission of the data in application area and/or send buffer.

For a SEND request without WAIT, DFHZARL then:

1. If there is sufficient space in the send buffer for all the data, transfers the data from the application area to the send buffer, and returns control to the caller.

2. Saves the INVITE and LAST indicators.

3. If the send buffer cannot hold all the data, invokes DFHZSDL for an implicit SEND.

For both cases, DFHZARL then:

1. Checks for a signal received.

2. Checks for exception response received. If so, calls DFHZERH to handle the error. On return, sets the state machine.

3. Returns to the caller.

When an implicit send is required, DFHZARL passes the data to DFHZSDL for transmission, passing the address of the data in the send buffer and in the application buffer. The total length of data passed to DFHZSDL is a multiple of the request unit size. On return to DFHZARL, the remaining data is transferred to the send buffer.

The parameters passed to DFHZARL, such as INVITE and LAST, are not transmitted by DFHZSDL.

RECEIVE Function

An LL receive is a request for one LLID record (one structured field). A BUFFER receive is a request for a given data length, irrespective of the number of structured fields. An end-chain indicator received before the data count has been reached terminates the request.

DFHZARL performs the following actions:

1. Sets the user state machine.

2. Transfers any data already in the receive buffer into the application area. If sufficient data to satisfy the request, or if INVITE, CONFIRM, or LAST has already been received, then returns to the caller.

3. Determines whether an LL or a BUFFER receive was requested.

4. Invokes DFHZRVL via the activate chain to issue the receive request.

   The parameters passed are:

   - Data length required (if known)
   - Address of receive buffer
   - LL or BUFFER receive indicator.

   The parameters returned are:

   - Length of data
   - Data flow control (DFC) indicators:

     CD

     CEB

     FMH

     RQD2

     Exception response.

DFHZRVL:

- Sets the user state machine accordingly.
- Updates the LL count.
- Transfers the data to the application area.
- Recalls DFHZRVL if more data required, or
- Returns to DFHZARL.

If errors are discovered following a receive request, DFHZRVL may return an exception response or an FMH 7. DFHZERH is invoked to handle these errors, and the user machine is set.

ISSUE ERROR/ABEND Function

DFHZARL is called as a result of an ISSUE ERROR or ISSUE ABEND command, and performs the following actions:

1. Sets the user state machine.
2. Calls DFHZERH.

DFHZARM

DFHZARM is invoked via the DFHLUCM macro.

This macro has seven executable options:

DFHLUCM TYPE =

- SEND
- RECEIVE
- WAIT
- SIGNAL
- FREE
- FLUSH
- INVALID_ID

DFHLUCM TYPE=STORAGE defines the storage in LIFO for passing primary input and output. The DSECT name is DFHLUMDS. TCTTE contains the secondary input and output. The principal functions are described in the following sections.

SEND Function

DFHZARM performs the following actions:

1. Maps the data into GDS format.

   The IDs used are:

   - X'12F1'
   - X'12F2'
   - X'12FF'

2. Examines bits set in the TCTTE by DFHZARQ to determine which DFC to apply.
3. Invokes DFHZARL (using the DFHLUC macro):

   - The first call passes the LLID.
   - The second call passes the address and length of data.
   - If necessary to concatenate data, further calls are made.
   - DFC requests go with the last call to DFHZARL.

4. Updates the state bits in TCTTE as necessary.
5. Interrogates the LU6.2 ATTACH_FMH_BUILT bit in the TCTTE, which was set by DFHZSUP or DFHETL. This bit indicates whether this is first SEND. If an LU6.2 attach header has not already been built as a result of a CONNECT PROCESS command, DFHZARM issues CONNECT_PROCESS to DFHZARL, assuming sync level 2, before sending the data.

RECEIVE Function

DFHZARM performs the following actions:

1. Calls DFHZARL using TYPE=BUFFER. Two calls are made. On the first call, the first 4 bytes (LLID) are retrieved into LIFO. These are examined and the LL is used to determine the TIOA size and to specify the length required in the second call.

2. On the second call, retrieves the remainder of the data directly into the TIOA. If the LL indicates concatenated data, then a series of calls is made to retrieve all the data.

RECEIVE in SEND State

If the application is in the send state and issues a RECEIVE command, DFHZARM causes null data (that is, LLID = X'000412F1') to flow with CD, if there is no data left to transmit. Otherwise, the remaining data is transmitted with CD.

FREE Function

The FREE function is used, for example, by DFHZISP to ensure that I/O has completed and CEB sent, using null data if necessary.

FLUSH Function

The FLUSH function is used, for example, by DFHSPP to ensure that any data deferred by DFHZARQ is sent before sync point flows are initiated. If necessary, null data is used.

INVALID_ID Function

The INVALID_ID function is used by DFHETL and DFHZARM itself. It handles the receipt of unrecognized or unsupported IDs. DFHZARM calls DFHZARL with ISSUE_ERROR (X'0889010x'), and sends a record with ID X'12F4' followed by the unrecognized ID. If the remote system responds, DFHZARM turns the flows around so that the local system can try again.

LU6.1 Chains

An LU6.1 chain corresponds to one SEND command. LU6.2 chains are bigger, so:

- For outbound data, DFHZARM maps one SEND into one structured field (concatenated if necessary).

- For inbound data, DFHZARM retrieves one (possibly concatenated) field and calls it a chain, thus preserving compatibility.

DFHZSDL

DFHZSDL transmits:

- Data from a send buffer and/or an application area

- The commands:

  LUSTATUS

  RTR

  BIS

- Responses.

Data Transmission

For data transmission, DFHZSDL uses the following VTAM Version 2 Release 1 enhancements:

- Large message performance enhancement outbound (LMPEO)

  – VTAM slices large messages into RUs.

- Buffer list (BUFFLST)

  – VTAM accepts data from non-contiguous buffers.

- User request header (USERRH)

  – The request header is passed in BUFFLST.

A maximum of two buffer list entries are used. The first buffer list entry addresses the data in the send buffer, and the second the data in the application area.

The request header is built in the first buffer list entry using parameters passed from DFHZARL. If an implicit send was requested, then CD, RQD2, and CEB are not checked. The first-in-chain (FIC) indicator is set after checking the chain state machine, and last-in-chain (LIC) set whenever CD, RQD2, or CEB is included. Null data sent only-in-chain (OIC) is converted to a LUSTAT6 command. The address of the send exit, DFHZSLX, is stored in the RPL and the VTAM SEND macro issued. On completion of the SEND request, the bracket and chain state machines are set according to the DFC indicators. These state machines are used extensively by DFHZERH to determine the state of the session before executing an error request.

Command Transmission

The LUSTAT6 command is:

- Sent with CEB to terminate BIND_in_bracket.
- Sent for OIC with null data.
- Sent with BB, RQD1 to BID for bracket.

The BIS command indicates bracket termination prior to CLSDST.

The RTR command requests BB after BID reject with sense code X'0814'.

On completion of the SEND request, the exit DFHZSLX is invoked. LUSTAT causes the bracket and chain state machines to be set as for normal data flow.

Response Transmission

DFHZSDL transmits ER1 and DR2 responses. The sequence number associated with the response is that of the path information unit (PIU) that initiated the current bracket. The response is sent synchronously, and POST = SCHED included in the VTAM command, so that an exit routine is not involved. On return from VTAM, DFHZSDL sets the bracket and chain state machines accordingly.

DFHZRVL

DFHZRVL is invoked to receive:

- Data
- Commands
- Responses
- Purge to end-chain (used by DFHZERH to flush incoming data.)
- A single RU.

The receive buffer is set up to receive the data, and the exit DFHZRLX is driven upon completion of the request. DFHZRVL is used by DFHZERH to determine the state of the session.

Data Received

When data is received, DFHZRVL:

1. Sets the bracket and chain state machines, and returns indicators to DFHZARL according to the DFC flags received with the data:
   - Response type
   - CD
   - EC
   - CEB
   - FMH.
2. Recalls DFHZRVL to receive further data if required.
3. Returns control to DFHZARL when:
   - Sufficient data has been received for a BUFFER or LL type request.
   - End-chain has been received due to either CD, RQD2, or CEB.
   - FMH has been received.

- The call was incomplete, but insufficient space remains in the receive buffer for further data.

If the data was received with RQD1, then a response is sent synchronously by DFHZRLX.

Command Received

When a command is received, the actions of DFHZRVL depend on the command:

- For LUSTAT6 received, the command is treated as data. If BB is included, then an exception response is sent (sense X'0813' or X'0814').
- For BIS received, CLSDST is requested and the receive redriven.

All other commands are invalid.

Response Received

When a response is received, DFHZRVL:

1. Performs checks:
   - Does the sequence number match the number of the BB request?
   - If it is a definite response, was it expected?
   - If it is an exception response, was it a session-level error?
2. Sets the state machines.
3. Passes back the return code to the caller.

DFHZSLX

This module is the VTAM exit which is driven on completion of a SEND request. If the request completed successfully, the bracket and chain state machines are set to indicate the new state of the session.

DFHZRLX

This exit is scheduled on completion of a RECEIVE SPECIFIC request. The request is checked for successful completion, and examined to determine whether data, a command, or a response has been received. Parameters indicating what was received are returned to the caller.

DFHZERH

Outbound Errors

For outbound errors, DFHZERH is invoked by DFHZARL following an ISSUE_ERROR, or ISSUE_ABEND, or SYNC_ROLLBACK request.

An FMH 7 needs to be transmitted, but that can only be done if the session is in the send state.

If the session is in the receive state, then DFHZERH:

1. Sends a negative response.
2. Purges the remaining data to end of chain.

In all cases, DFHZERH then:

1. Checks that the session is still in bracket.
2. Flushes the send buffers.
3. Calls DFHZARL to send the FMH 7.

Inbound Errors

For inbound errors, DFHZERH is invoked by DFHZARL when either a process-level exception response or an FMH 7 has been received.

If an exception response is received, the session must be manipulated into receive state in order to receive the following FMH 7. It is necessary to flush the present output chain followed by a dummy FMH 7 with the INVITE option. DFHZARL is then called to receive the incoming FMH 7.

If an FMH 7 is received, DFHZERH examines the associated sense code and any GDS error log data, then returns to DFHZARL.

DFHLUP and DFHZLUS

These modules form the LU services manager, discussed in the following sections.

Session Management – The LU Services Manager

Class of Service: LU6.2 makes use of the class of service facility in VTAM. The TCT structure for LU6.2 reflects this. Under the system entry (TCTSE) are a series of mode group entries (TCTME). Within a mode group there are a number of sessions represented by terminal entries (TCTTE).

All the sessions within a mode group have the same transmission characteristics, that is, the same class of service. When a request to ALLOCATE a session is made, the MODENAME is specified, indicating which class of service is required.

When there are parallel sessions between two LU6.2 systems, it is possible to vary the number of sessions available using master terminal commands, either globally, or by MODEGROUP. The two systems must agree on numbers at any time. LU6.2 defines a transaction which runs in each system, communicating with its counterpart and controlling the number of sessions available. This transaction is known as the "LU services manager." Commands to CHANGE_NUMBER_OF_SESSIONS (CNOS) are exchanged between the LU services managers. In the event of resynchronization being required, the LU services manager performs EXCHANGE_LOG_NAME (XLN) to ensure that the correct restart is possible.

So that the LU services managers can always communicate with each other, even when all the sessions between two systems are busy, two extra sessions are always created whenever parallel sessions exist between two systems. CICS generates these two extra sessions (with a reserved MODENAME of SNASVCMG) whenever FEATURE=PARALLEL is specified for the TCTSE. No other transaction is allowed to use these two sessions.

LU Services Manager: The transaction processing names (TPNs) defined by LU6.2 for the LU services manager functions are:

- X'06F1' = CHANGE_NUMBER_OF_SESSIONS (CNOS)
- X'06F2' = EXCHANGE_LOG_NAME (XLN)

The PCT definitions required are:

TRANSID	XTRANID	PROGRAM
CLS1	X'06F10000'	DFHLUP
CLS2	X'06F20000'	DFHLUP

These are generated automatically if DFHPCT TYPE=GROUP, FN=(ISC) is specified.

Whenever LU services manager operations are required, DFHLUP is invoked. DFHLUP calls DFHZLUS or DFHSPP to perform the requested function.

DFHLUP

In the local CICS system, DFHLUP is invoked using the DFHILUS macro. This macro sets up an IC START command (specifying TRANSID=CLS1) and records data on temporary storage indicating the requested operations. The DFHLUS operations can be:

- INIT_CH_SESS – Initialize
- CH_SESS – Change sessions
- SHUTDOWN – Shutdown
- RESYNC – Resynchronize.

DFHLUP retrieves the temporary storage record, and, for the first three operations, calls DFHZLUS. For the RESYNC operation, DFHSPP is called.

When an ATTACH request is received by the LU services manager from a remote system, DFHLUP is attached. DFHLUP examines the receive buffer area and for:

- TPN = X'06F1' (CNOS), calls DFHZLUS
- TPN = X'06F2' (XLN), calls DFHSPP.

DFHZLUS

DFHZLUS handles all requests to change the number of sessions. It is invoked using the DFHLUS TYPE=Z macro. The possible calls are:

- When CICS is the sender:
  - INIT_CH_SESS – Initialize
  - SHUTDOWN – Shutdown
  - CH_SESS – Change sessions
- When CICS is the receiver:
  - REC_CH_SESS – Receive change sessions.

Initialization

DFHLUP is started by the DFHZNAC exit routine for DFHZOPX, and DFHZLUS is called by DFHLUP. DFHZLUS performs the following actions:

1. Issues a GETMAIN to obtain a buffer to hold one CNOS command.
2. Does a privileged allocate.
3. Builds an attach header.
4. Completes the building of the CNOS command, using MAXSESS values in the TCTME.
5. Issues a SEND INVITE WAIT.
6. Issues a RECEIVE LLID.
7. Analyzes the responses to the command – CICS must accept the negotiated values.
8. Sets new active values for the TCTME. If CONNECT=AUTO is specified in the TCTME, all contention winners are bound at this time.
9. Sends one or more messages.
10. Frees the session.

If another group is to be initialized, repeat the above steps.

Shutdown

DFHLUP is STARTed by DFHZSHU, and DFHZLUS is called by DFHLUP.

DFHZLUS performs the following actions:

1. Issues a GETMAIN to obtain a buffer to hold one CNOS command.
2. Does a privileged allocate.
3. Builds an attach header.
4. Completes the building of one CNOS command, setting MAX, WIN and LOS values to zero, and mode names affected to ALL.
5. Issues SEND INVITE WAIT.
6. Issues RECEIVE LLID.
7. Analyzes the response to the command – the responder must accept zero sessions (DRAIN can be changed from ALL to NONE).
8. Sets new active values for each TCTME.
9. Sends a message.
10. Frees the session.

Change Sessions

DFHLUP is started by DFHEMD (CEMT), and DFHLUS is called by DFHLUP. If the whole system is changed, for example by CEMT SET SYSTEM (...) ACQUIRED|RELEASED, initialization or shutdown is performed. CEMT SET VTAM CLO also causes shutdown.

DFHZLUS performs the following actions:

1. Issues a GETMAIN to obtain a buffer to hold one CNOS command.
2. Does a privileged allocate.
3. Builds an attach header.
4. Completes the building of one CNOS command, setting MAX and WIN values.
5. Issues SEND INVITE WAIT.
6. Issues RECEIVE LLID.
7. Analyzes the response to the command – CICS must accept the negotiated values. If CONNECT = AUTO is specified in the TCTME, all new contention winners are bound at this time.
8. Sends one or more messages.
9. Frees the session.

Receive Change Sessions

DFHLUP is ATTACHed, and calls DFHZLUS. DFHZLUS performs the following actions:

1. Addresses the input buffer.
2. For each mode group selected, determines whether the values for MAX and WIN are acceptable – if not, amends the CNOS command in the buffer.
3. Sets LOS, that is, this system's contention winners, to the lower of either the remaining sessions (MAX_SESSIONS – the other system's winners) or this system's maximum contention winner count.
4. If this system is currently performing shutdown, negotiates down to zero.

Exchange Log Name

When the sync point program (DFHSPP) requires to resynchronize interrupted units of work or simply exchange log names with a remote system, it invokes the LU services manager, using the DFHLUS macro and specifying RESYNC. This starts DFHLUP (with its own TCA). When DFHLUP sees that resynchronization is required, it invokes DFHSPP (still under the new TCA) to perform the function.

When TPN X'06F2' is received from a remote system, DFHLUP invokes DFHSPP to handle the operation.

SYNC POINT and RECOVERY

SYNC LEVEL 1

Sync level 1 is implemented in CICS using the CONFIRM option on the SEND command, and the ISSUE CONFIRMATION command in response. SEND CONFIRM causes any outstanding data to be sent with RQD2. The application program then waits for the response. ISSUE CONFIRMATION causes DR2 to be sent in response. This is the only effect of these commands. The CICS sync point mechanisms are not involved in any way.

SYNC LEVEL 2

If a conversation has been started at sync level 2, then whenever CICS sync pointing takes place, as a result of EXEC CICS SYNCPOINT, task termination, and so on, the session carrying on the conversation participates in the sync point.

Sync Point Flows

Sync point requests are communicated to the remote system via SPS headers. There are four commands:

- Prepare to commit (PTC)
- Request commit (RC)
- Committed

- Forget.

The flows are shown in Figure 164. System A is the initiator and systems M1 and M2 are secondary systems. The state of the participating systems at each stage is shown in brackets, for example, **(in-doubt)**.

After sending RC, a node does not know, until it sees the reply, whether to commit or backout. Some systems (for example, CICS) cannot hold locks indefinitely and must commit or backout, even when the reply is not forthcoming. This is known as **heuristic commit** or **heuristic backout**.

The actions taken by CICS are determined by the DTB parameter in the PCT macro:

- DTB = YES – heuristic backout
- DTB = (YES, NO) – heuristic commit
- DTB = (YES, WAIT) – wait for resync if possible.

For LU6.2, resynchronization is attempted on any parallel session before a heuristic action is taken.

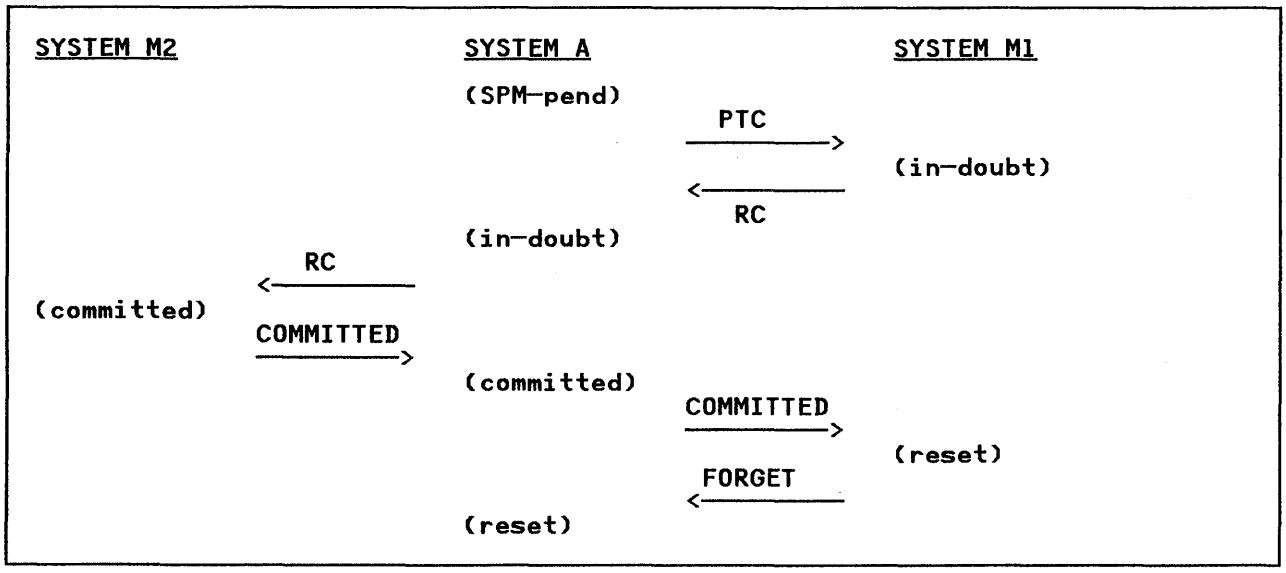


Figure 164. LU6.2 Sync Point Flows

Unit of Recovery Descriptor

The unit of recovery descriptor (URD):

- Maintains the state ready for resynchronization.
- Is recoverable via emergency restart.
- Contains:
  - DISPOSITION (SPM-PEND, INDOUBT, and so on)
  - SYSIDNT
  - A pointer to waiting resources
  - Message inserts
  - Unit of work ID (UOWID)
  - Conversation correlator
  - Session instance identifier.
- URDs are addressed from the TCTTE while in use. They are also in the URD chain for activity keypointing and while awaiting resynchronization.

Unit of Work ID (UOWID)

The UOWID shown in Figure 165 on page 340 is passed in the LU6.2 attach header when a conversation is initiated. Subsequently, sync point processing causes the sequence field to be increased each time a successful sync point occurs.

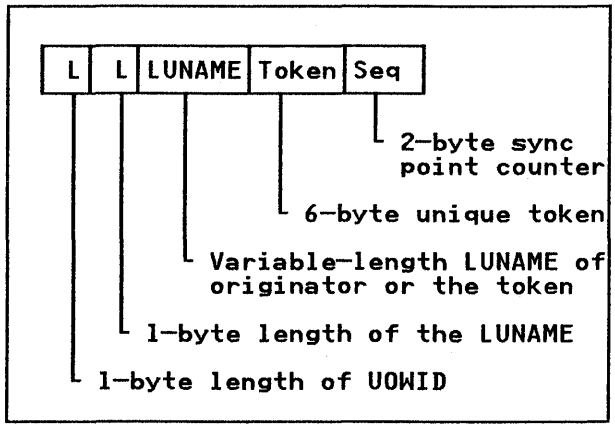


Figure 165. Unit of Work ID

Conversation Correlator

The conversation correlator is passed in the LU6.2 attach header when the conversation is initiated. It is stored in the TCTTE LU6.2 extension. CICS uses the TCTTE name for the conversation correlator when building an attach header.

TCTTEs and URDs

Unlike LU6.1 sequence numbers, URDs are not associated with the original session (TCTTE) for resynchronization. URDs are pointed to by the CSA and the TCTTE as shown in Figure 166.

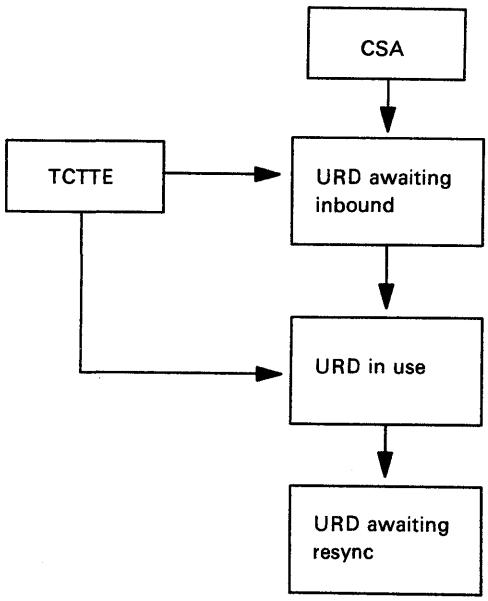


Figure 166. Unit of Recovery Descriptor

Session Instance Identifier

This is used to resolve problems that can arise when the two ends of a session are notified of failure at different times.

The session instance identifier is:

- Created at each BIND (not recoverable at emergency restart).
- Exchanged at BIND.
- Sent in COMPARE STATES (see "COMPARE STATES" later in this chapter).

The session instance identifier allows the COMPARE STATES receiver to synchronize on appropriate session outage notification (SON).

Log Name

LU6.2 systems communicating at sync level 2 must exchange log names before any communication can begin. If, subsequently, one system fails and is restarted, part of the resynchronization process requires the exchange of log names again to ensure that the systems have the same basis for synchronization.

The log name used is the middle 4 bytes of the start STCK taken at the last cold start and converted to hex.

EXCHANGE_LOG_NAME (XLN) is one of the functions of the LU services manager. TPN X'06F2' is used and the GDS structured field in Figure 167 on page 341 shows the log name information.

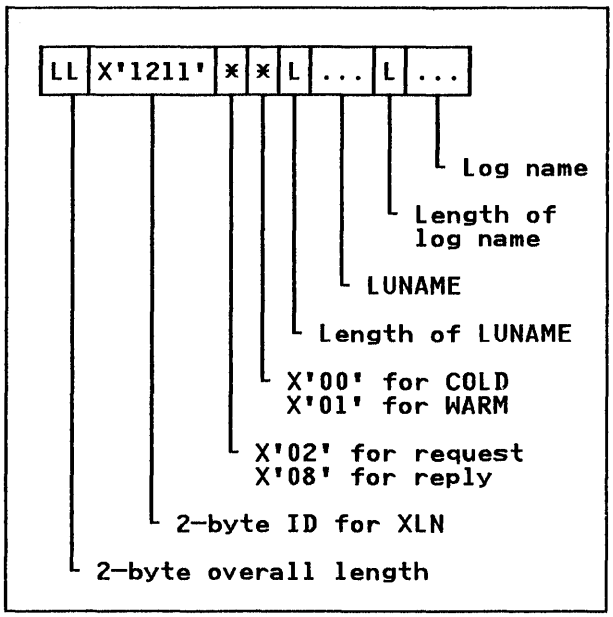


Figure 167. Log Name Information

In this context, COLD implies that the local system has no memory of the other system's log name. WARM means that the local system has memory of the other system's log name. The logs are deemed to match if, and only if, the systems start up in one of the following ways:

- Local system COLD, other system COLD.
- Local system COLD, other system WARM.
- Local system WARM, other system WARM, and its log matches the local CICS system's memory of it.
- Local system WARM, other system COLD, and the local CICS system has no URDs awaiting resynchronization.

Sync level 2 attaches are not allowed until successful XLN (this applies even without mismatch).

Following a log mismatch, subsequent attempts at resynchronization are inhibited until:

- A CICS restart, or
- The other system sends an acceptable log name (that is, it restarted), or
- The master terminal command CEMT SET CONNECTION(sysid) NOTPENDING is issued. This erases all URDs hanging off the system entry (with an integrity loss diagnostic) so that from then on CICS accepts any log name.

COMPARE STATES

At resynchronization, the LU service manager is asked to COMPARE STATES (CS). The GDS structured field shown in Figure 168 on page 342 holds the information.

After COMPARE STATES exchange, the combination of the two states is sufficient to ensure that both know where the other got to, and what to do to get back into step.

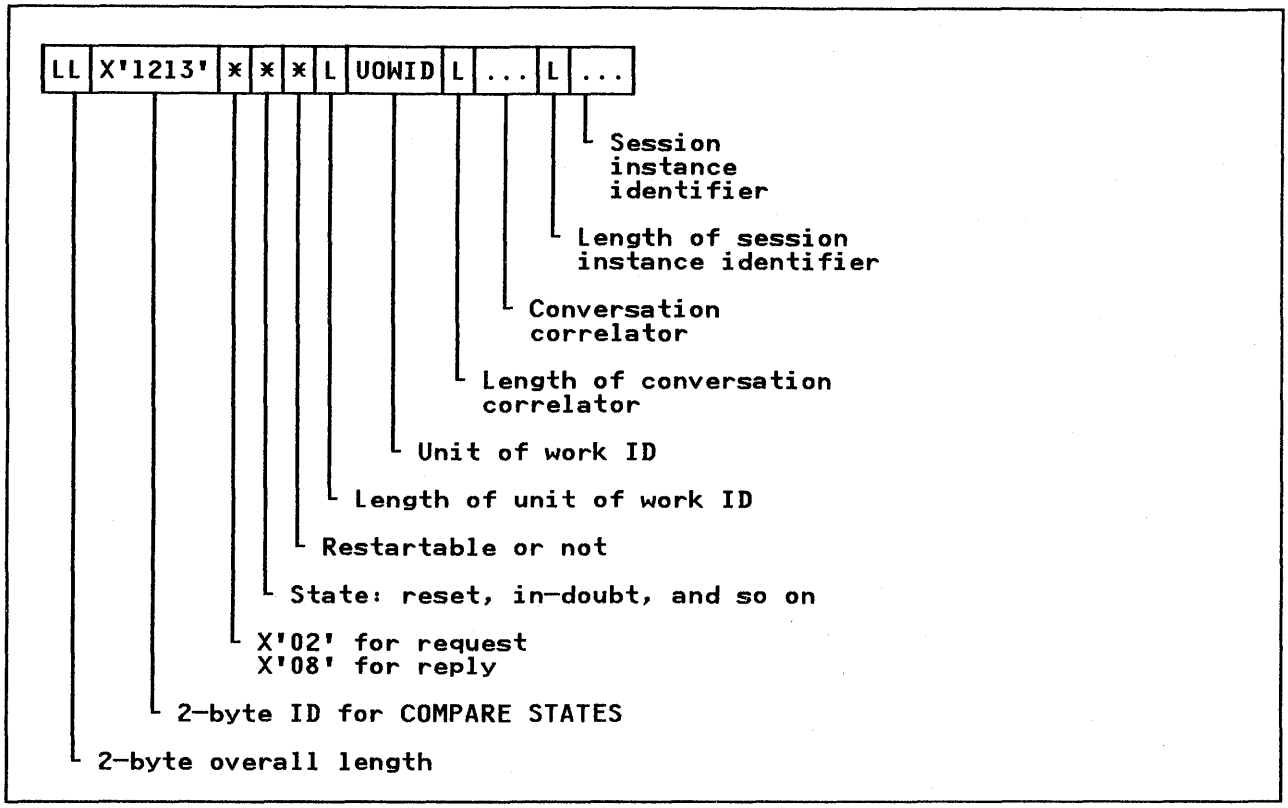


Figure 168. COMPARE STATES

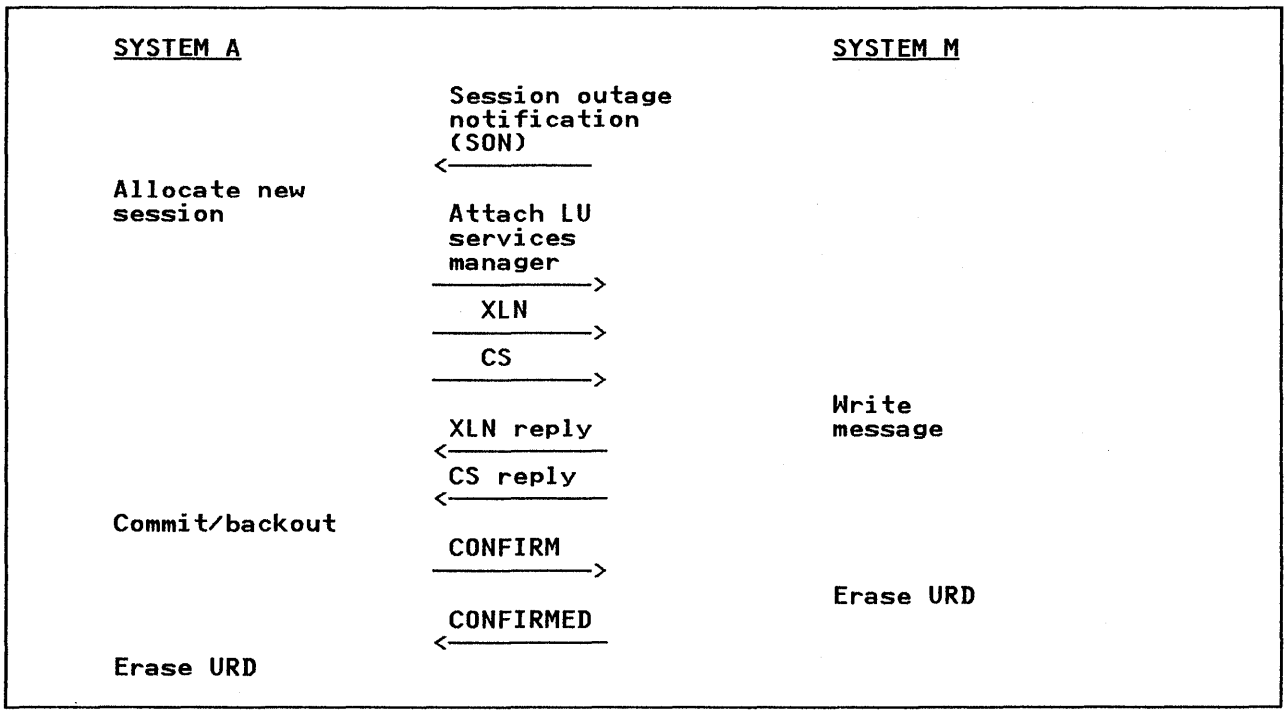


Figure 169. LU6.2 Resynchronization

Resynchronization

The flows shown in Figure 169 on page 342 take place when resynchronization is required. System A is the initiator and system M is a secondary system.

Resynchronization is initiated under the following circumstances:

- At BIND time when there are URDs awaiting resynchronization (see "Exchange Log Name" on page 338).
- At the first BIND after restart to establish log names, under the LU service manager's TCA.
- Following session failure during sync-point flows, in which case:
  - The sync-point initiator drives resynchronization flows under the sync-pointing TCA. Resources are not committed or backed out until resynchronization ends.
  - The sync-point secondary system abends the TCA with a message, commits, or backs out according to the point of failure and DTB specification, and holds the URD ready for inbound resynchronization.

System Failures

During normal processing, all UOW state changes are recorded on the system log (as well as in URDs). The URDs are keypointed to protect against log wraparound. Other systems' log names are recorded in special URDs which are also logged. Emergency restart recovers URDs from the log and marks them as awaiting resynchronization. Waiting resources are restored to that state, or heuristic decisions are taken where necessary. System failure then looks like many session failures and resynchronization is performed after the BIND without recognizing the difference.

Warm Shutdown/Restart

URDs are not recovered by WARM restart, so if there are URDs awaiting resynchronization, the restart data set is flagged so that RESTART=AUTO selects emergency restart. If no URDs are awaiting resynchronization, the CICS log name is saved and recovered to allow reconnection to a system expecting continuity.

Interfaces into DFHSPP

DFHSPP can be invoked by the following macros:

- DFHSP TYPE=CONTACT,SYSIDNT=

  invoked at first BIND, or at BIND when awaiting resynchronization.

- DFHSP TYPE=UNBIND,URDADDR=

  invoked at SON when there is a pending URD.

- DFHSP TYPE=RECXLN,SESDATA=

  invoked by the LU services manager when XLN is received.

- DFHSP TYPE=LUCRSY,SYSIDNT=,URDADDR=

  invoked by the LU services manager which was started by DFHSPP.

- DFHSP TYPE=QPEND,SYSIDNT=

  invoked by the CEMT INQUIRE PENDING command, or by DFHWKP during warm shutdown.

- DFHSP TYPE=CLPEND,SYSIDNT=

  invoked by the CEMT SET NOPENDING command.

Part 3. Organization

This part provides a brief description of the organization of all major CICS modules, and consists of CICS organization diagrams and tables containing alphabetically organized module information.

This part contains the following chapters:

- Chapter 3.1. Module Organization
- Chapter 3.2. CICS Executable Modules
- Chapter 3.3. Control Block Copybooks and Macros.

Chapter 3.1. Module Organization

CICS source-program listings are the key to the organization of CICS. You get to the listings from the organization diagrams (Figure 170 to Figure 199) and the individual module descriptions contained in the tables following the organization diagrams in this chapter. That is, you are aware of the general function being performed; that leads you to the organization diagrams (arranged by function) which identify specific modules involved in that function. Once you have located the module name that interests you, go to the tables containing the alphabetically organized list of the major modules comprising CICS. Armed with this information you are now ready to turn to a source-program listing to find the additional information you require.

CICS code is well commented. The processing steps within a module are described throughout the code. Many of the modules interact extensively with other CICS components during execution. Certain control blocks (the format and content of which are detailed in the *CICS/MVS Data Areas* manual) provide vital system information.

This chapter is designed to enable you to quickly index to the pertinent organization information on any CICS module.

CICS Organization Diagrams

The following diagrams show the major functions of CICS, each diagram being broken down by either module or subroutine name.

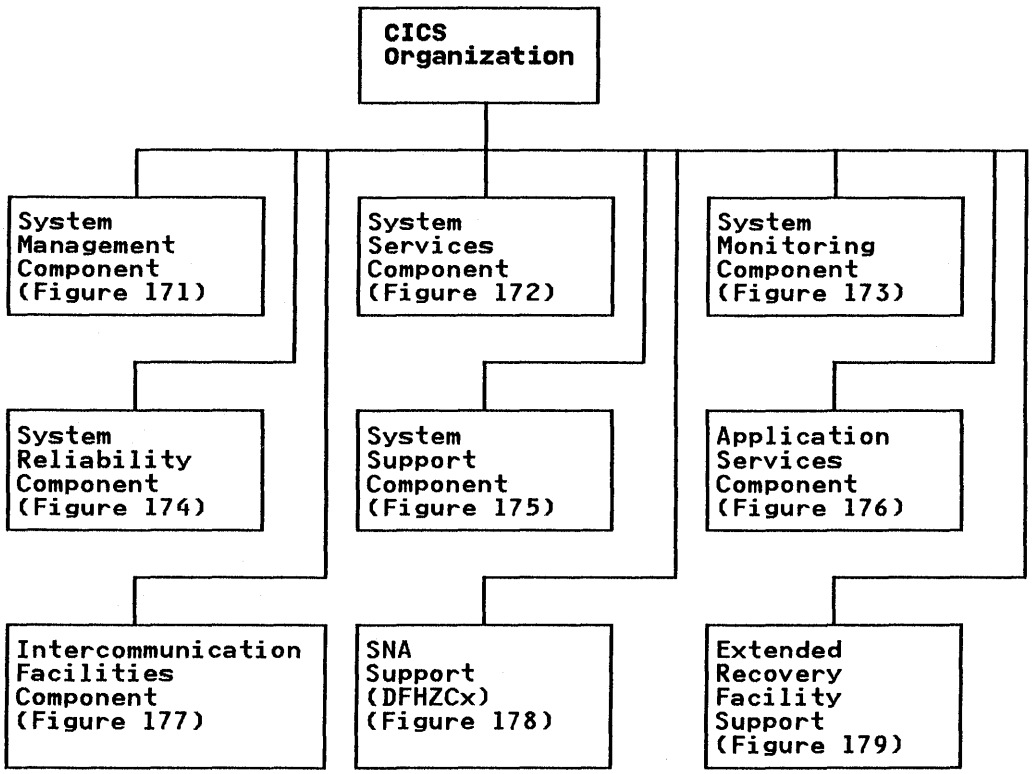


Figure 170. CICS Organization

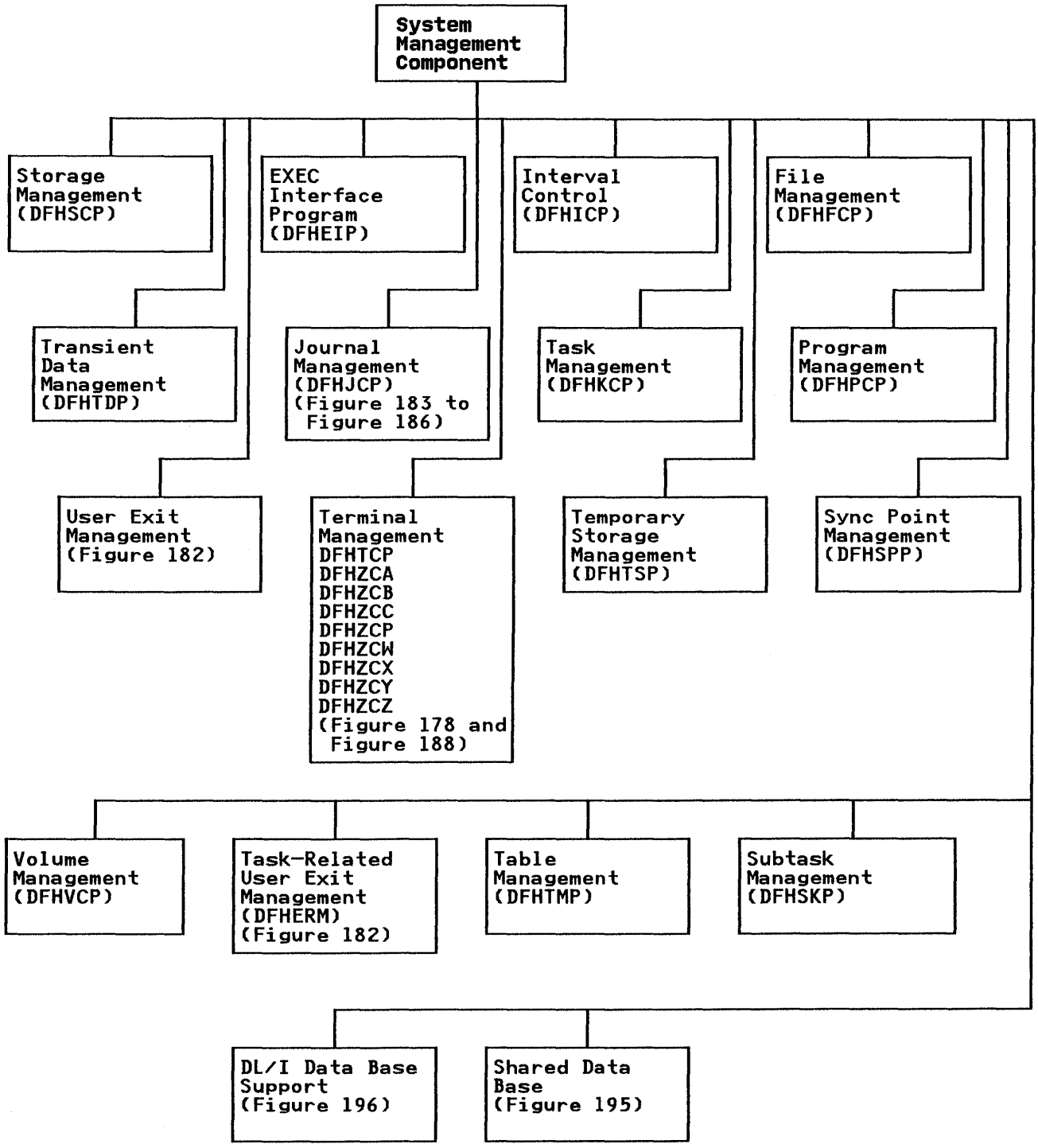


Figure 171. System Management Component

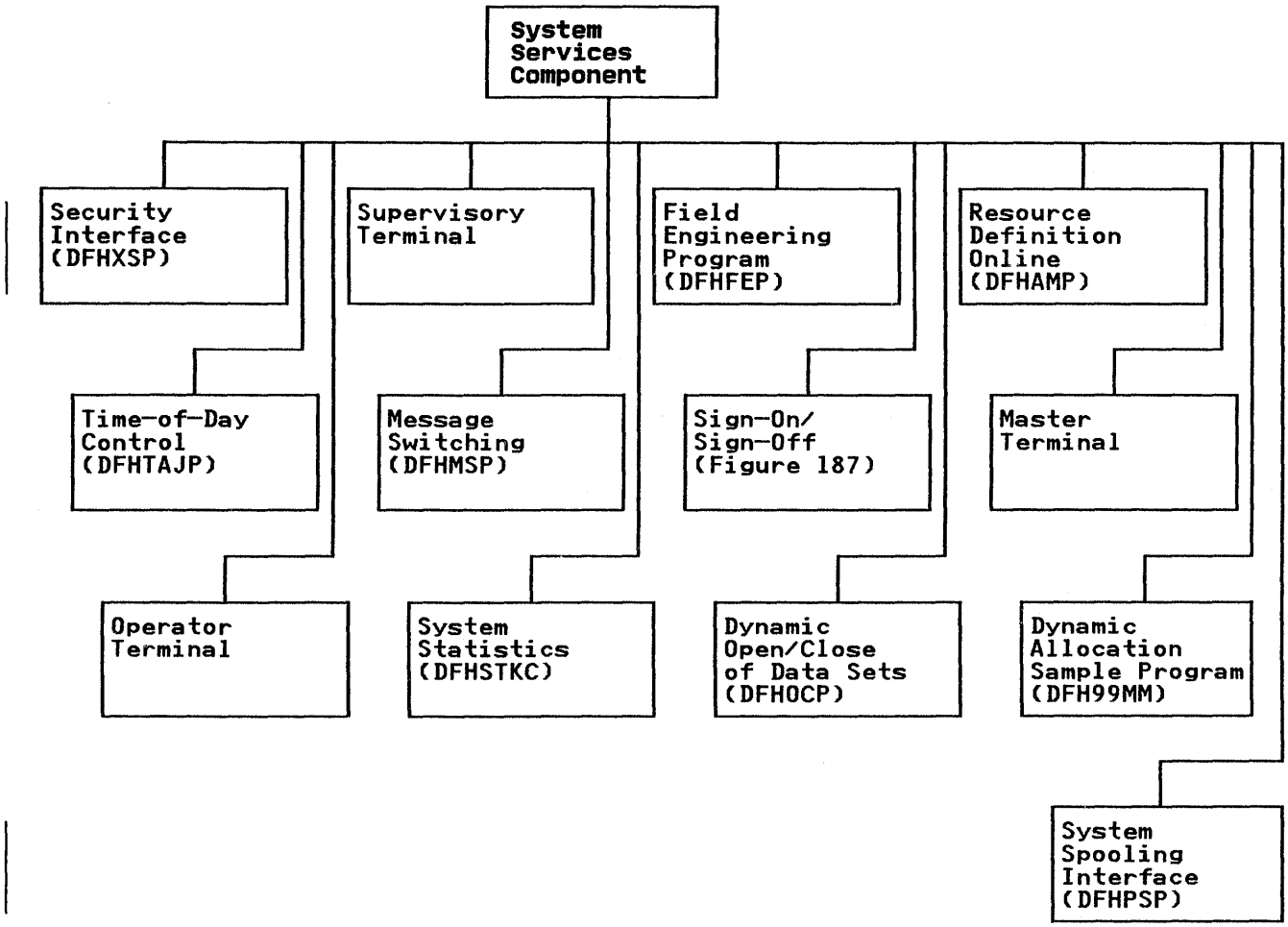


Figure 172. System Services Component

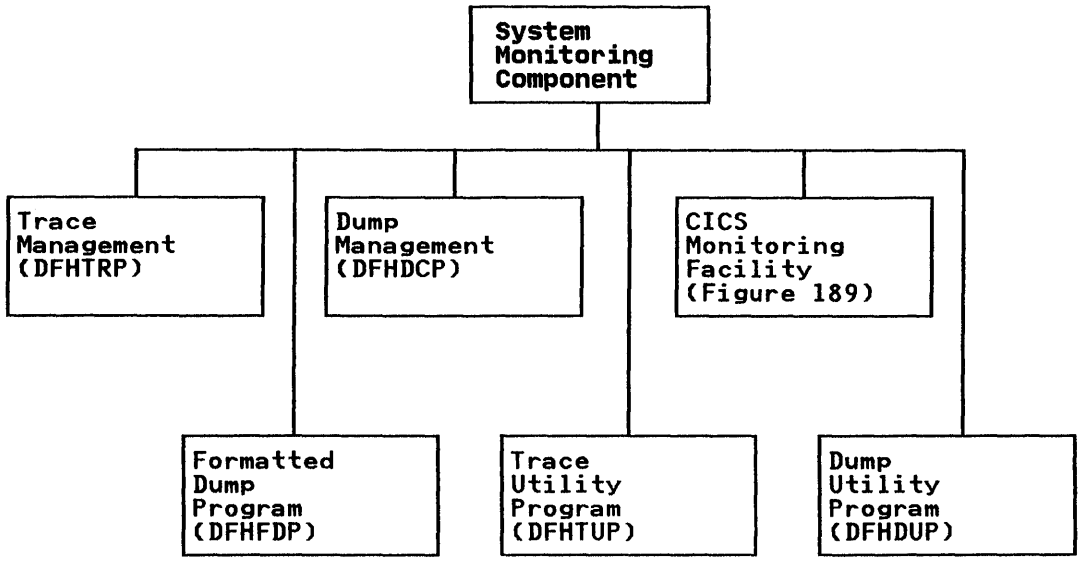


Figure 173. System Monitoring Component

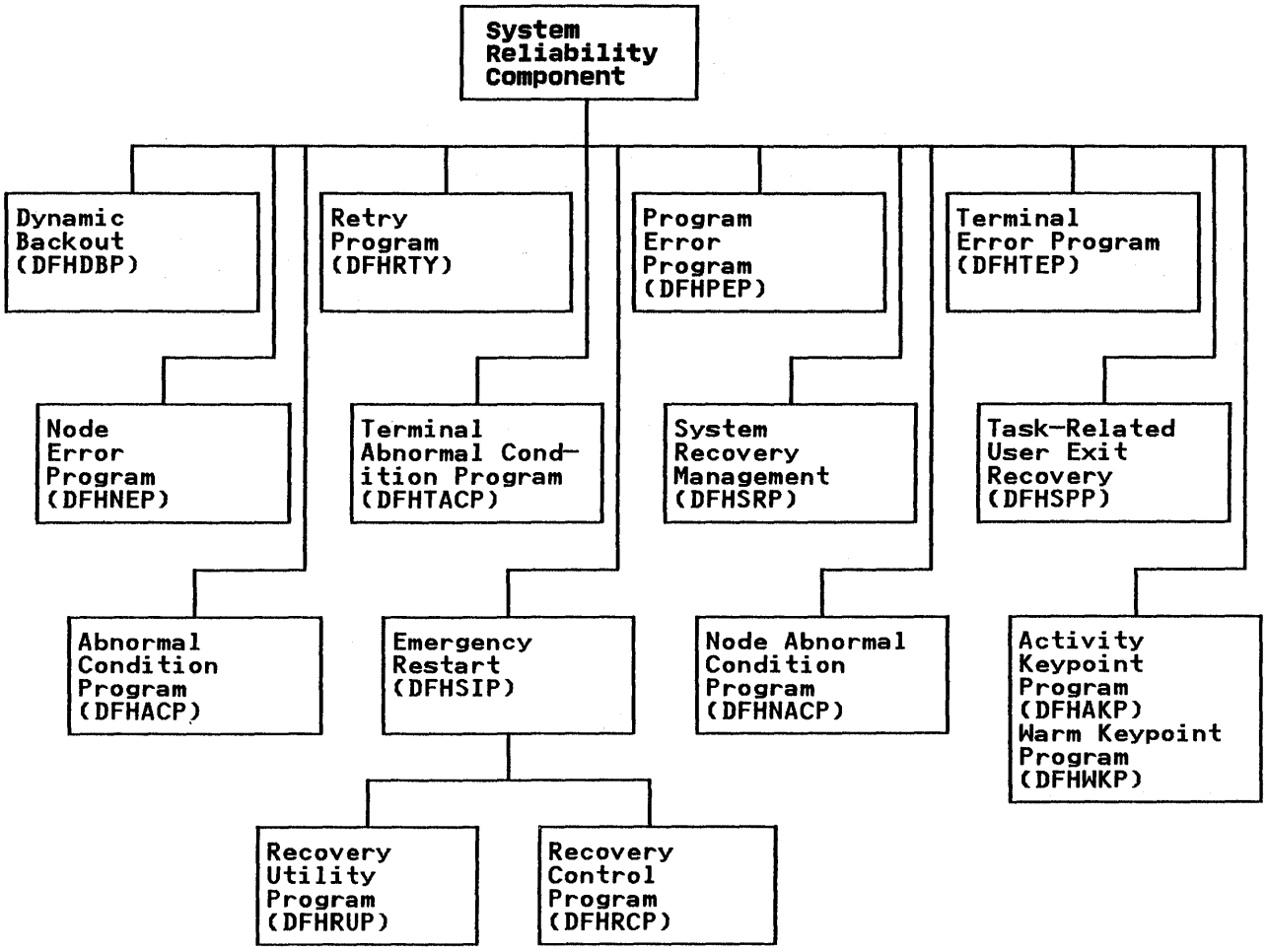


Figure 174. System Reliability Component

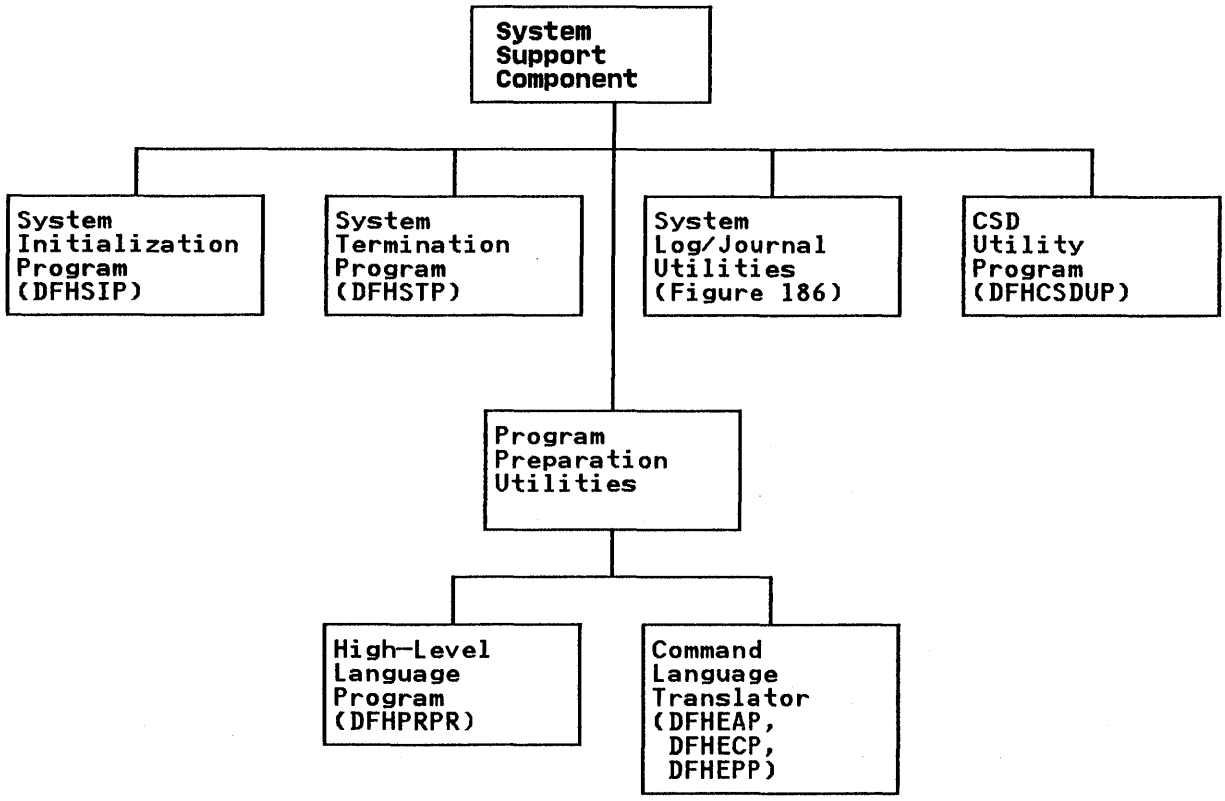


Figure 175. System Support Component

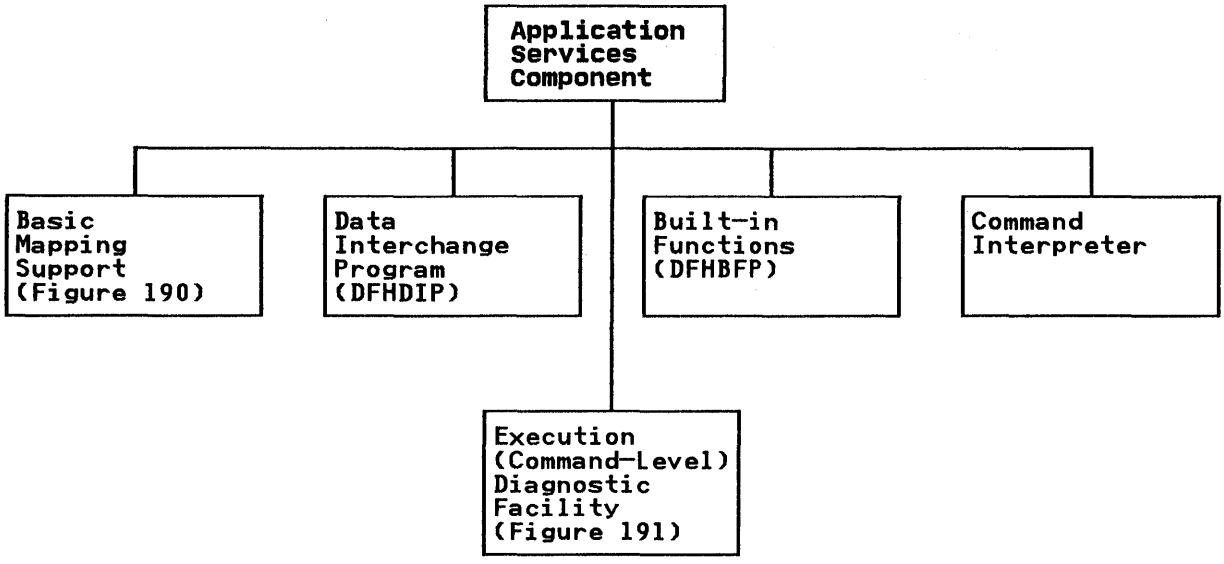


Figure 176. Application Services Component

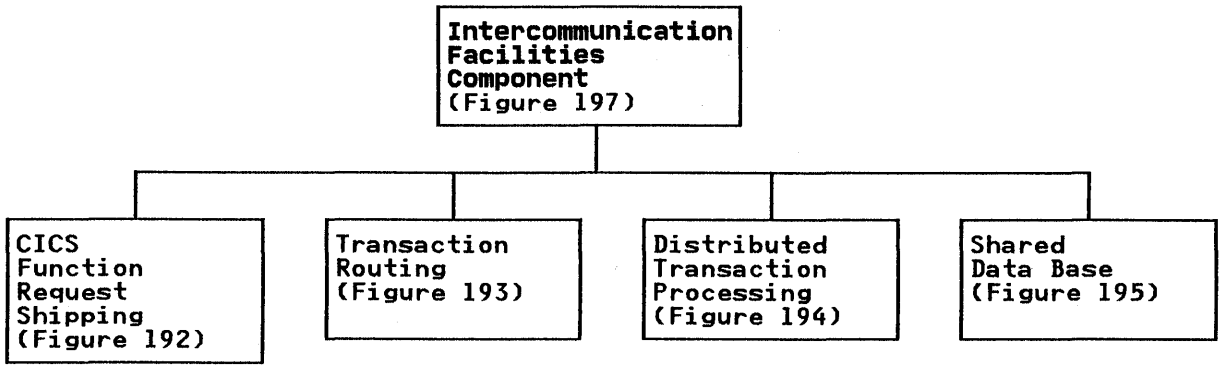


Figure 177. Intercommunication Facilities Component

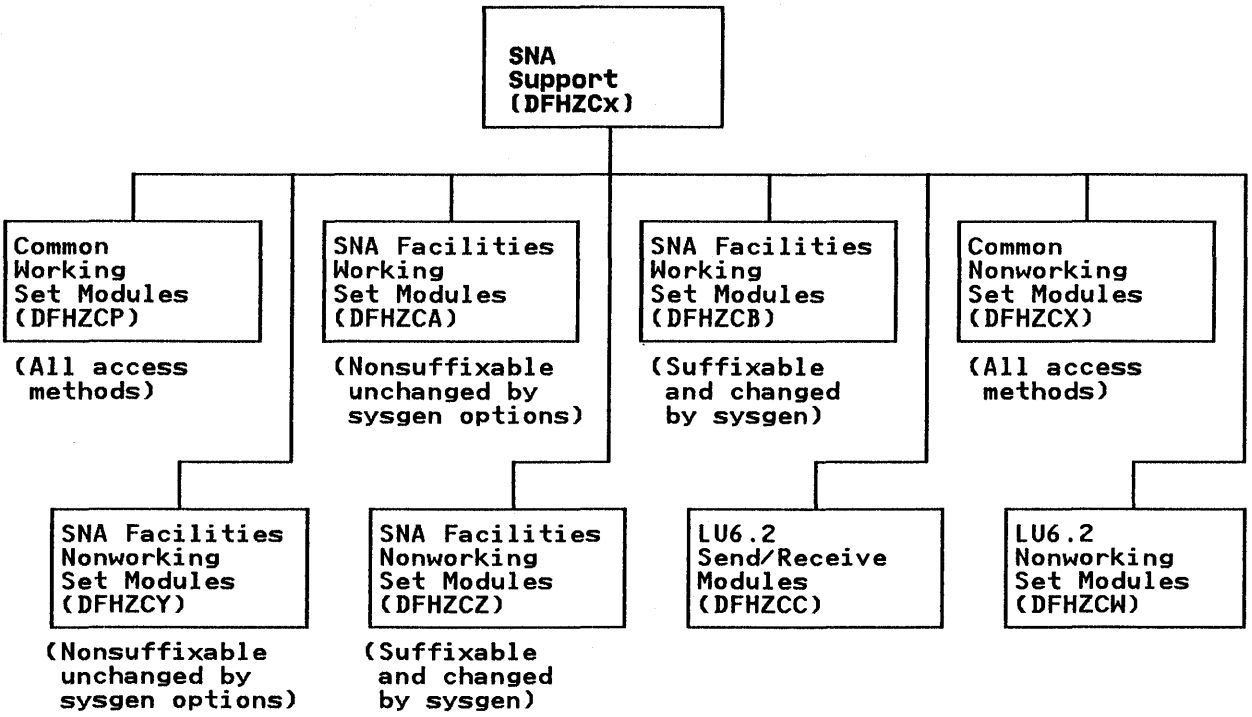


Figure 178. SNA Support (DFHZCx)

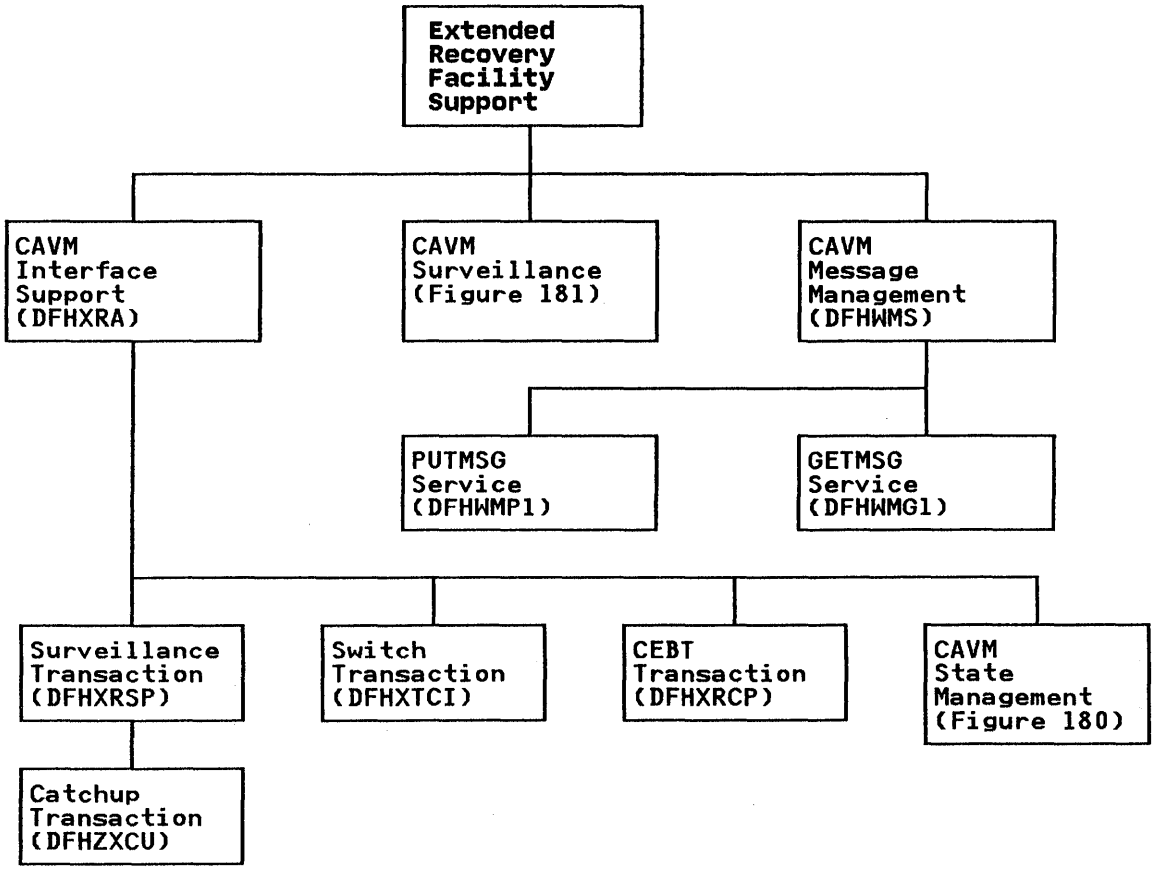


Figure 179. Extended Recovery Facility Support

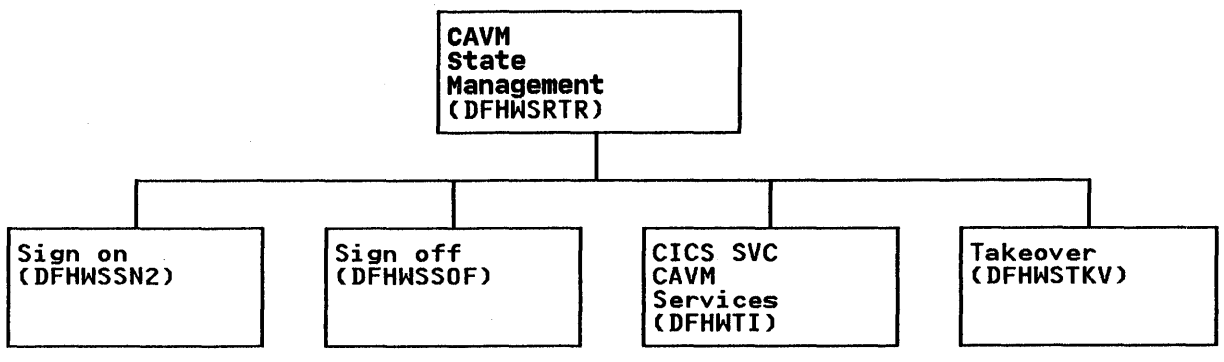


Figure 180. CAVM State Management

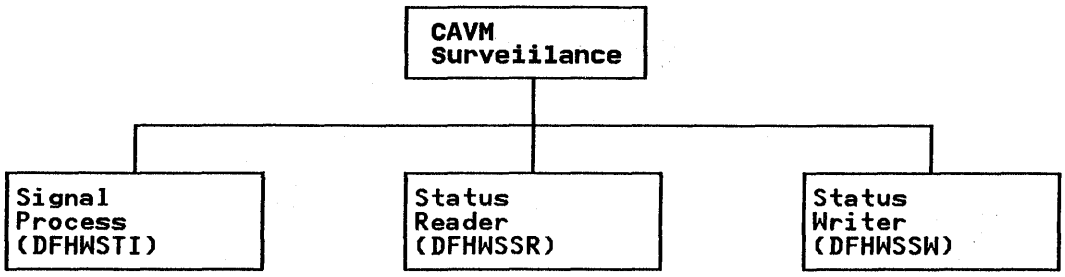


Figure 181. CAVM Surveillance

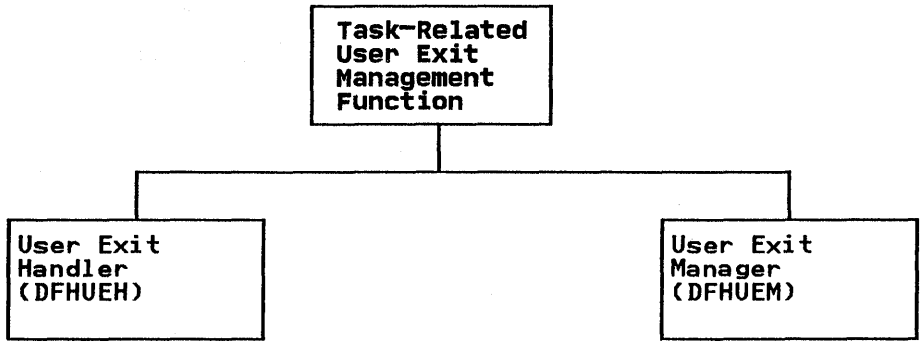


Figure 182. User Exit Management Function

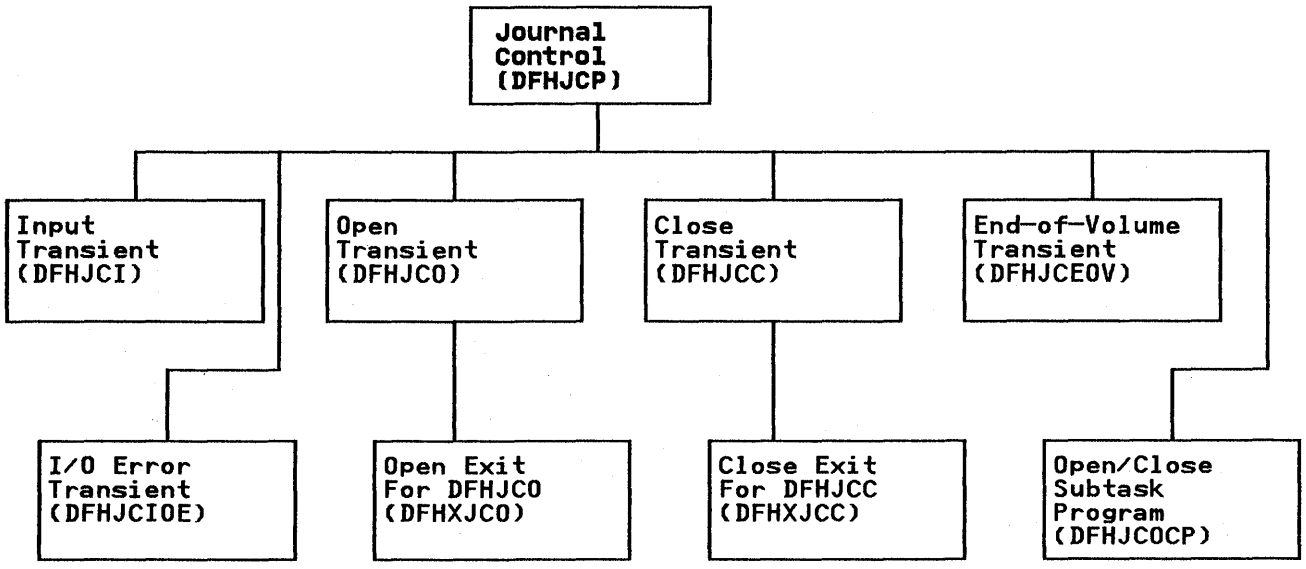


Figure 183. Journal Control

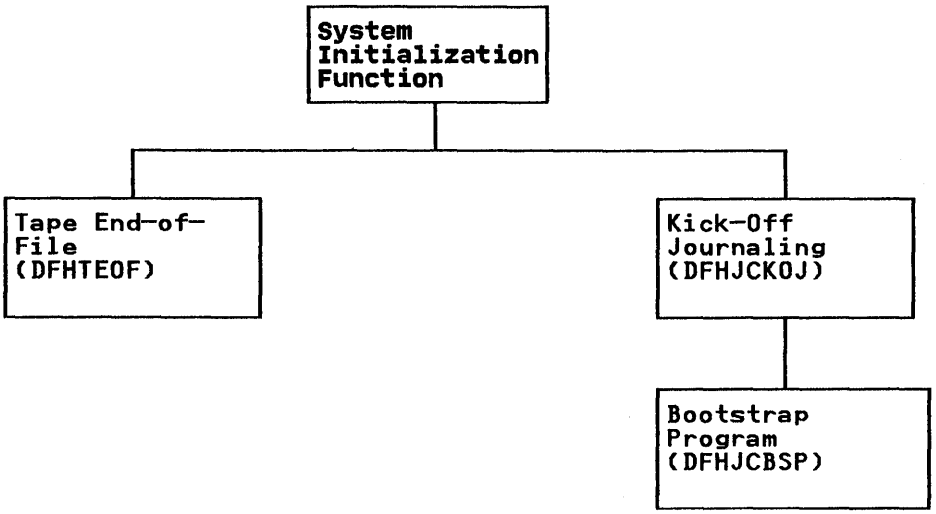


Figure 184. Journal Control – Initialization

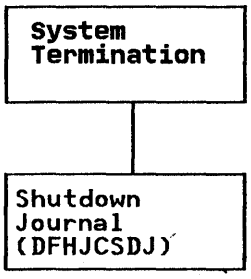


Figure 185. Journal Control – Termination

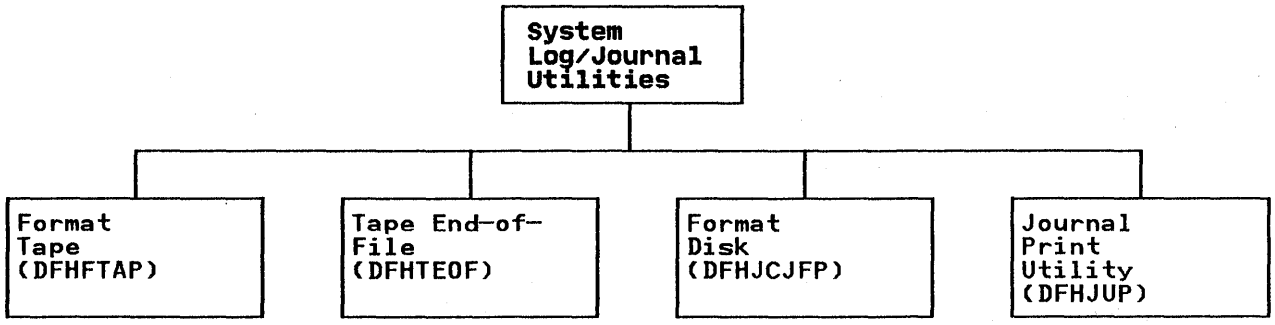


Figure 186. Journal Control – Offline Programs

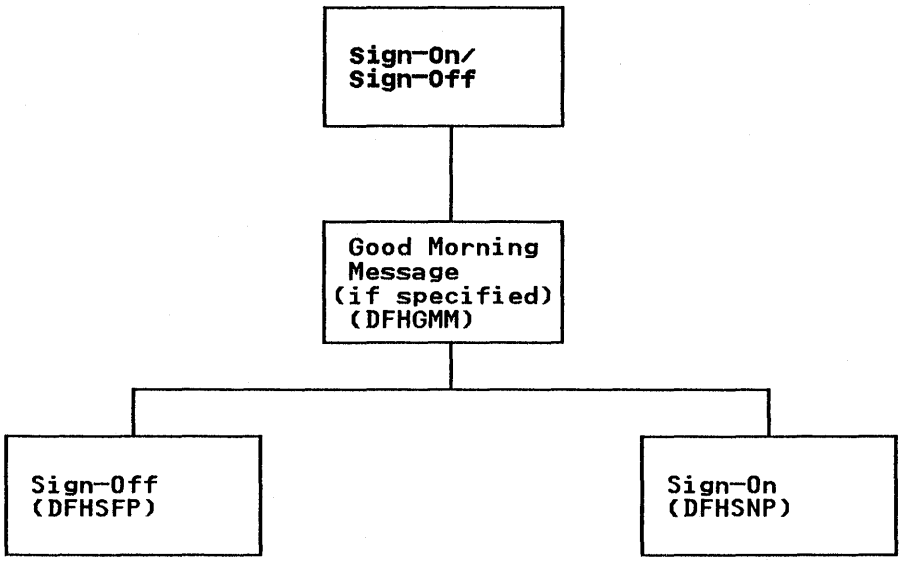


Figure 187. Sign-On/Sign-Off

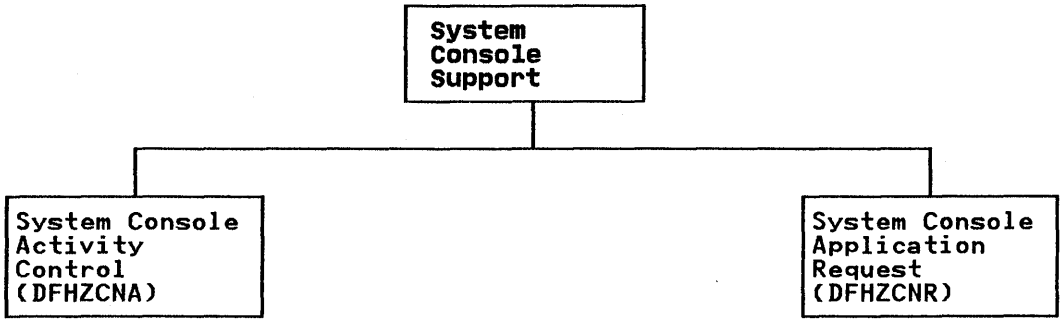


Figure 188. System Console Support

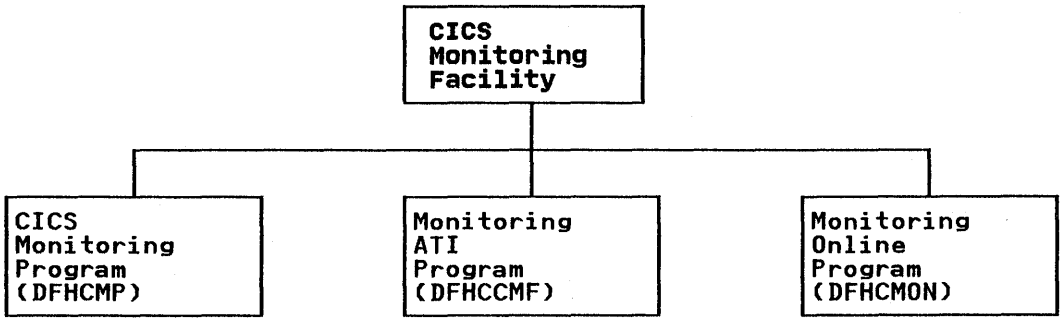


Figure 189. CICS Monitoring Facility

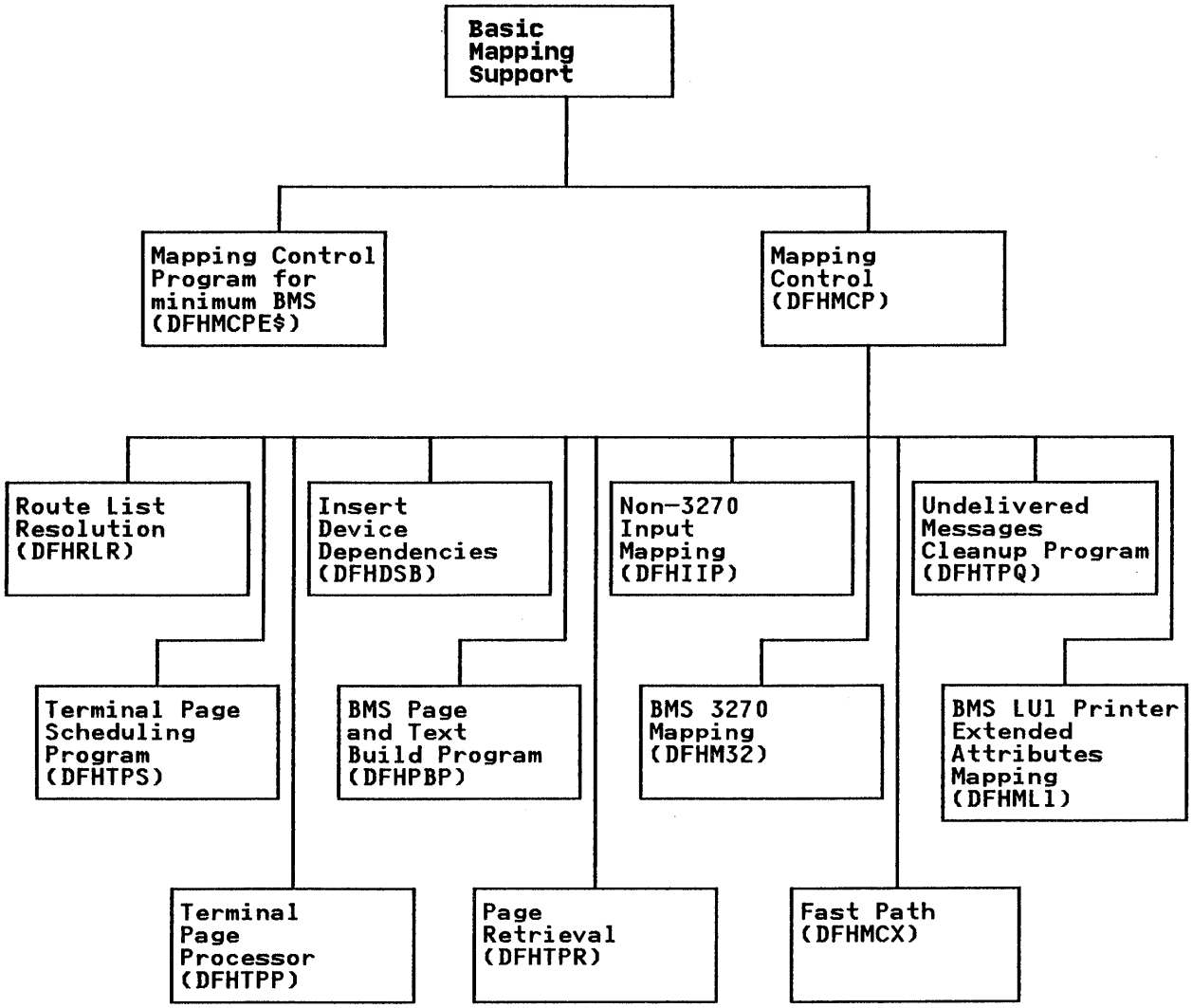


Figure 190. Basic Mapping Support

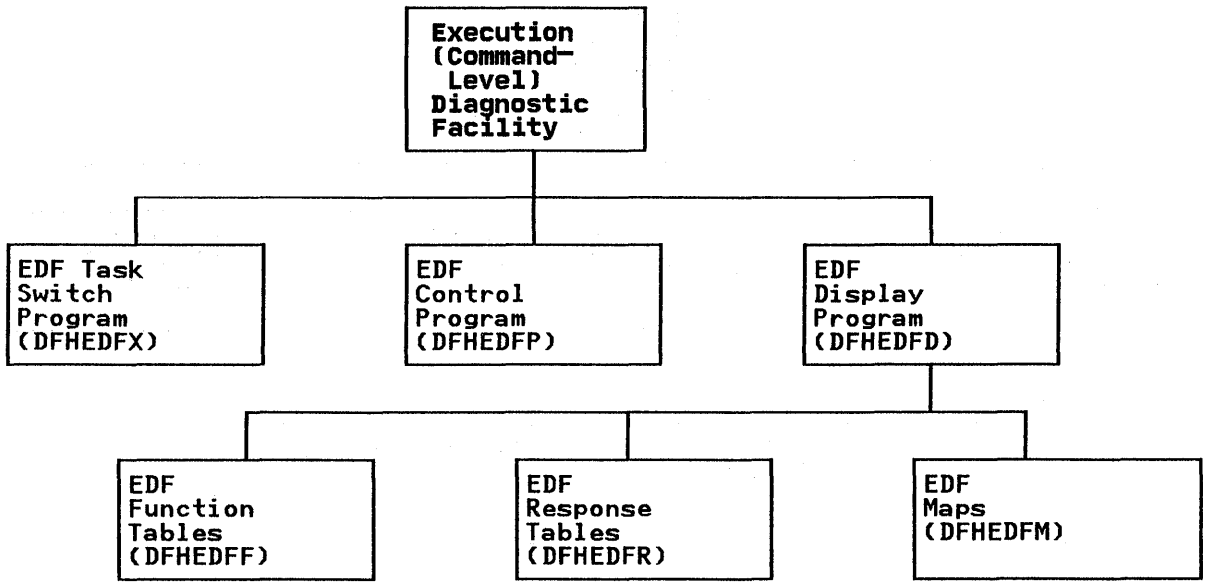


Figure 191. Execution (Command-Level) Diagnostic Facility

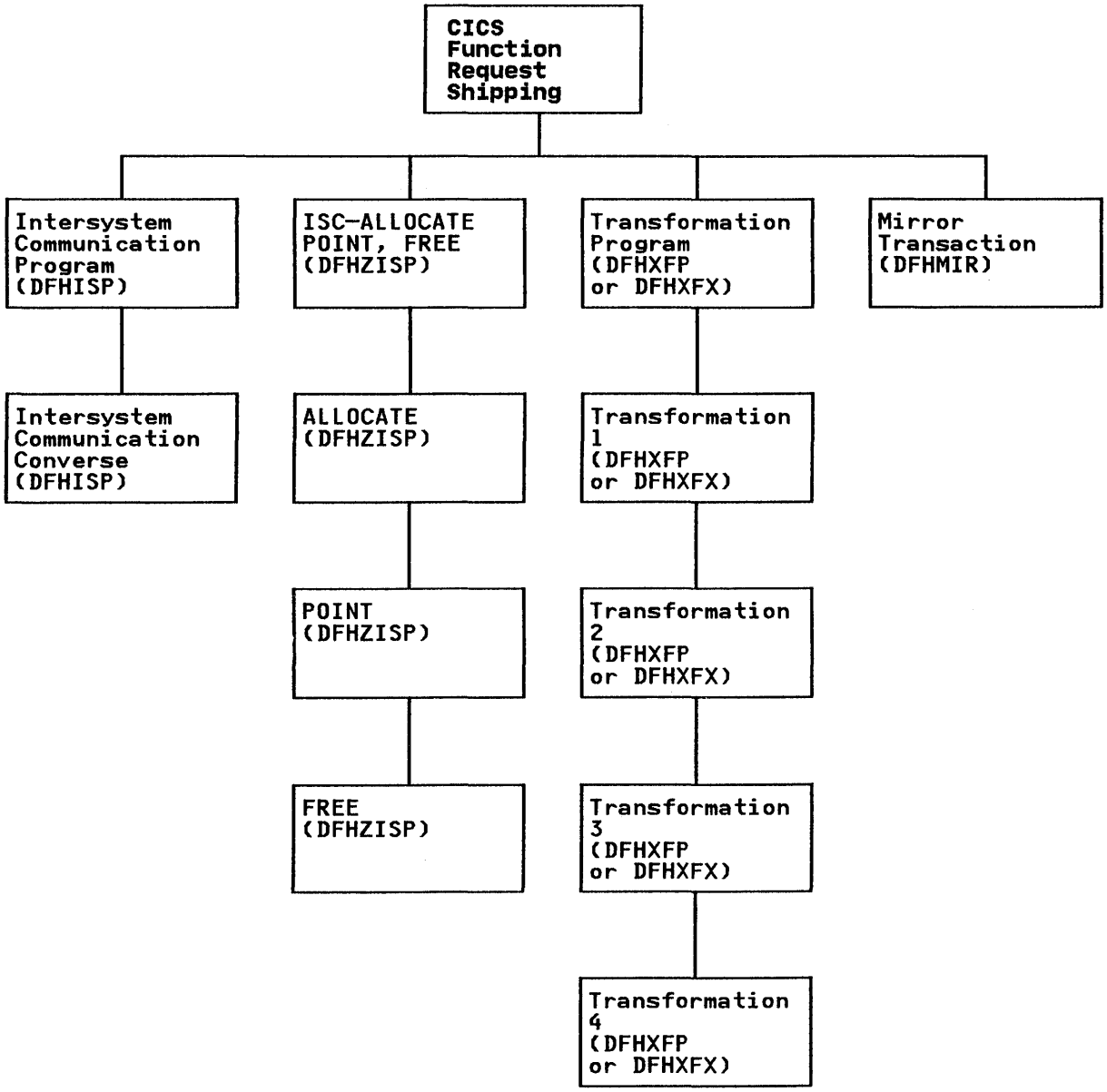


Figure 192. CICS Function Request Shipping

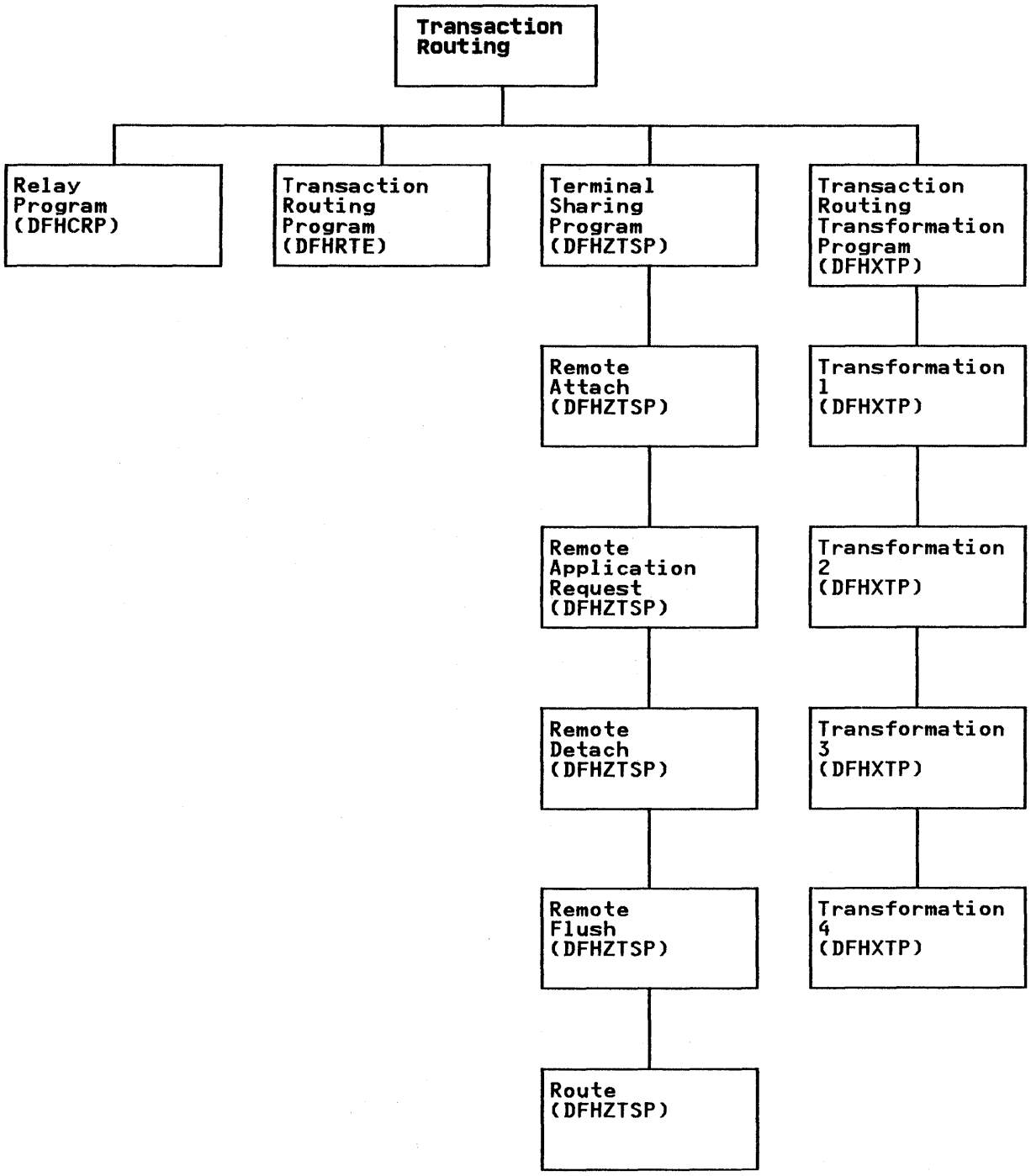


Figure 193. Transaction Routing

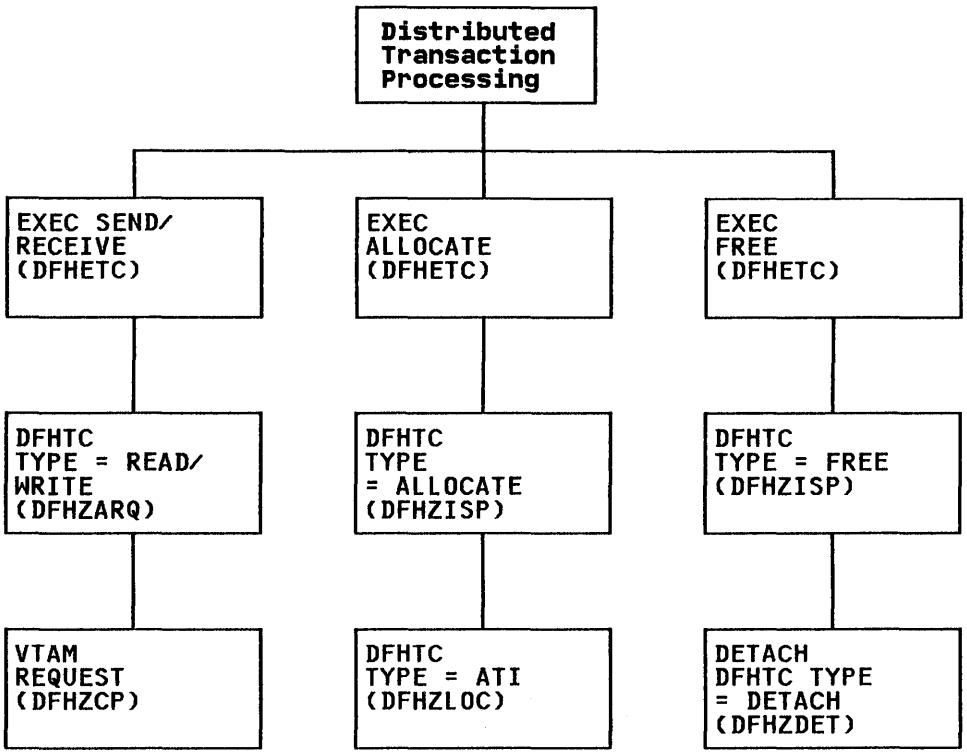


Figure 194. Distributed Transaction Processing for LU6.1

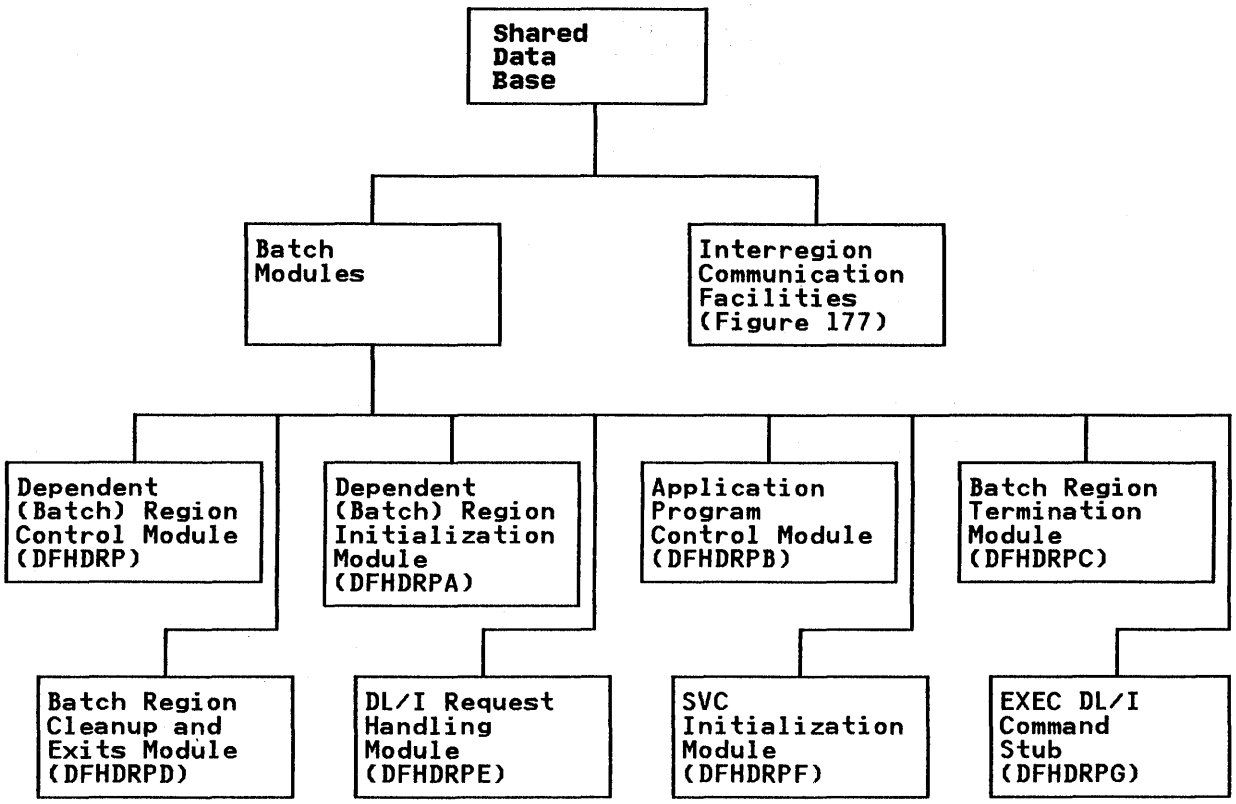


Figure 195. Shared Data Base

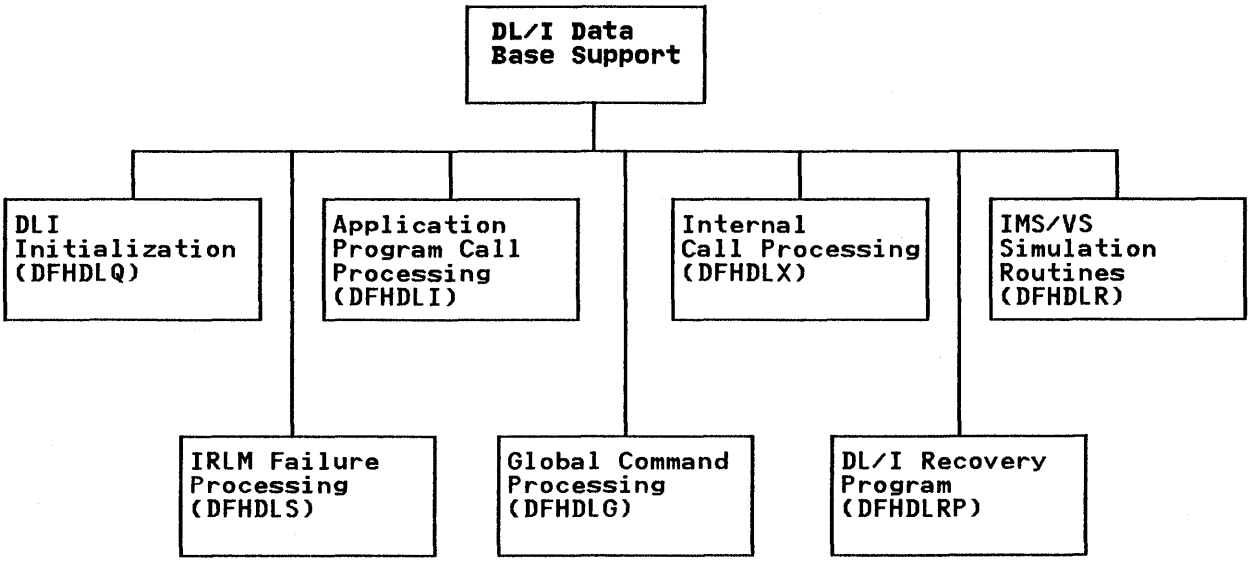


Figure 196. DL/I Data Base Support

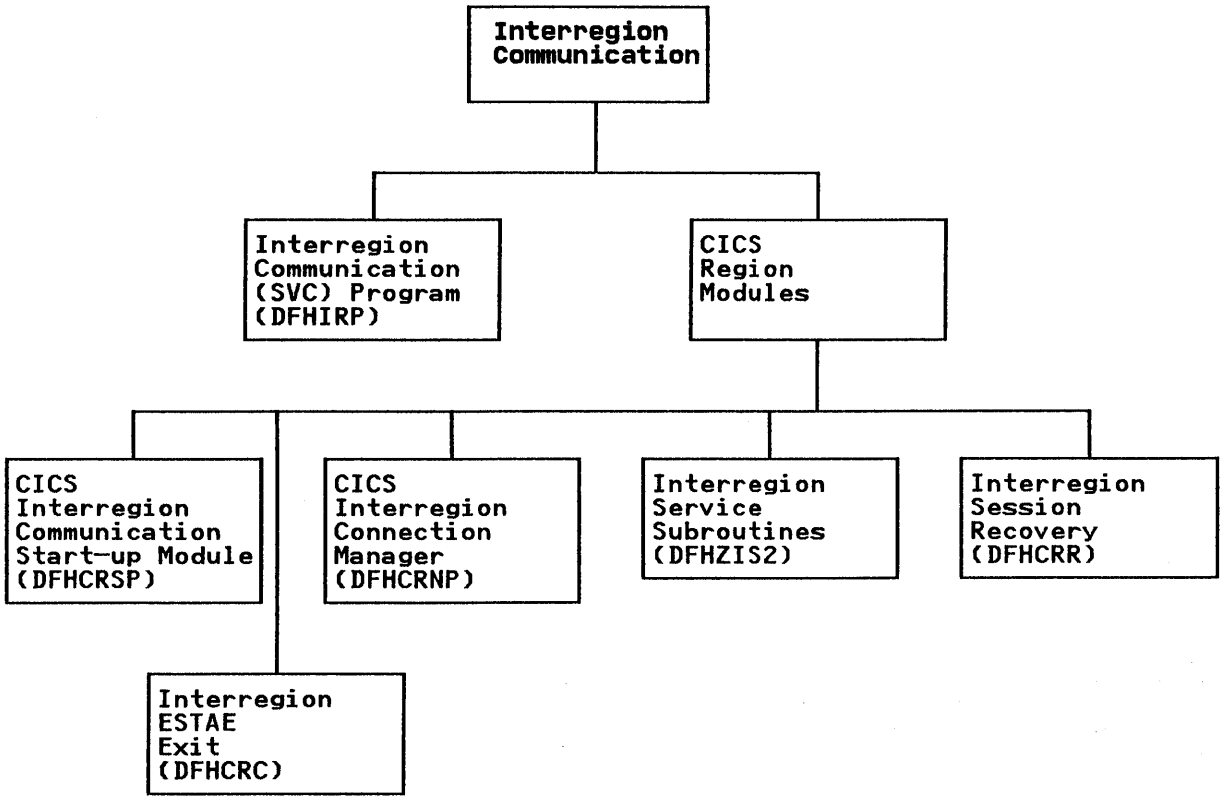


Figure 197. Interregion Communication

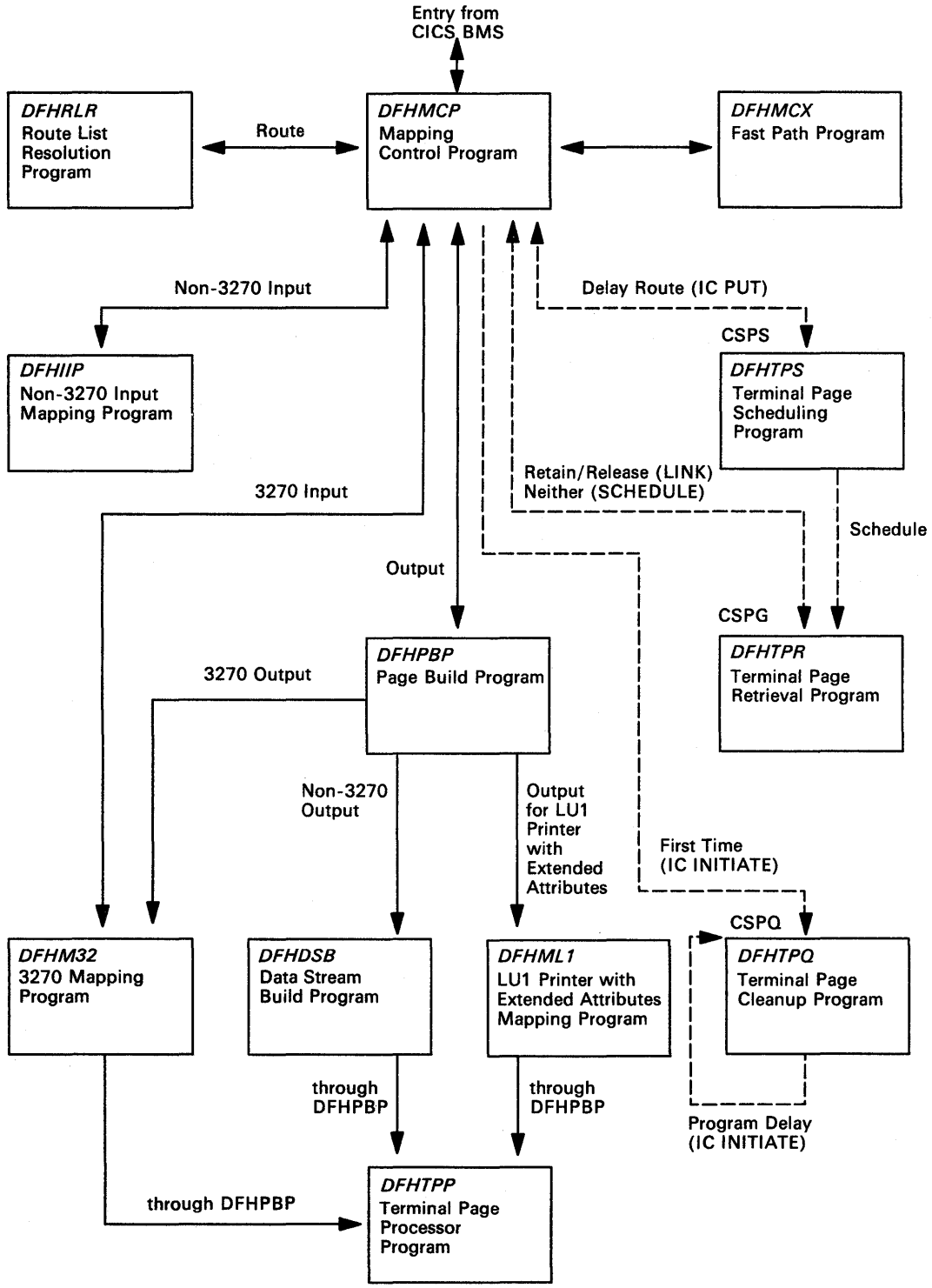


Figure 198. BMS Modules and Routines

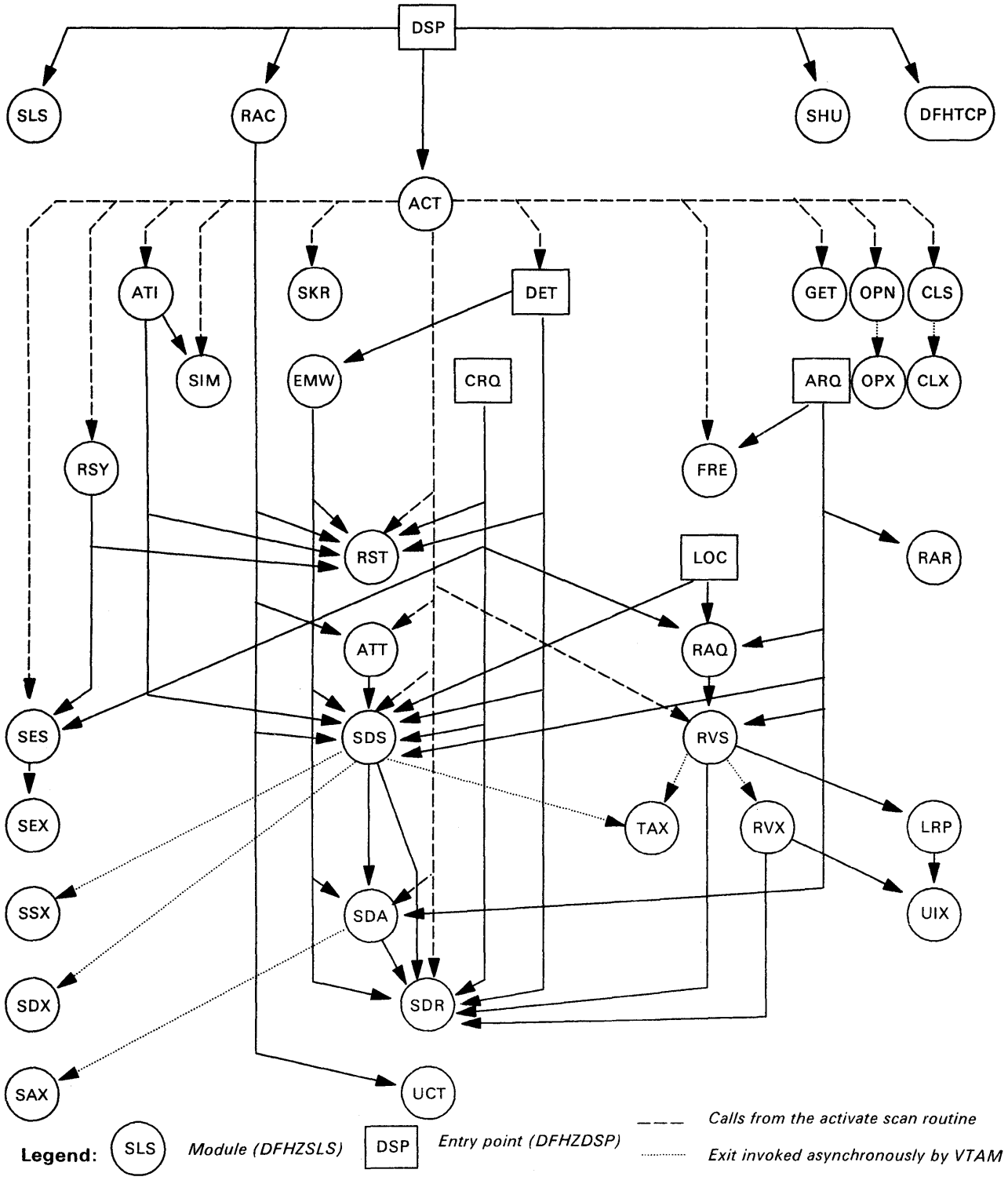


Figure 199. VTAM Terminal Management Control Flow

PROCESSING ROUTINES	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1
A	3																																
B																																	
C			2																														
D				2																													
E						2																											
F								2																									
G											2																						
H												2																					
I													2																				
J														2																			
K															2																		
L																					2												
M				3	3	3	3	5	3	3	3	5	3	3	3	3	3			4	5	3	2 3	3	3	3	3	3	3	3	3	4	4
N																																	
O		3																												2			
P																																	
Q	5		3		4						4			4	4		4														5	5
R		4					4	7	4	4		7	4			4				5	6	4	4	4	4	4	4	4	4	4	4		
S				5		6																											
T	2		1																														
U	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1	1	1	1	1
V					2																												
W								3				3																					
X								4				4								3	3												
Y																2																	
Z																	2																
AA																															2		
AB																																	
AC																																	
AD						4																											
AE		2																															
AF	4					5																											3
BA				4				6				6																			3		
BB																				2													
BC																						2											
BD																					4												
BE																								2									
BF																									2								
CA																										2							
CB									2																								
CC																											2						
CD																												2					
CE																													2				
CF																																	
DA										2																							
DB							2																										
DC																																	
DD																																	
DE																															2		
DF																																2	

Figure 200. TACP Message Construction Matrix

Note: The matrix shown in Figure 200 defines the selection of message routines based on error code. The sequence in which the routines are executed is indicated by the number in the column corresponding to the error code.

For example, for error code X'88', the processing routines are executed in the following order: U, F, W, X, N, BA, S. Refer to Figure 201 on page 370 for a list of the processing routines.

Routine	Function
A	Generate message "DFH2501 MSG TOO LONG, PLEASE RESUBMIT"
B	Not used
C	Generate message "DFH2503 AUTO OUTPUT HAS BEEN REQ. PLEASE PREPARE TO RECEIVE"
D	Generate message prefix "DFH2502 TCT SEARCH ERROR"
E	Generate message prefix "DFH2505 POLLING LIST ERROR"
F	Generate message prefix "DFH2507 INPUT EVENT REJ"
G	Generate message prefix "DFH2512 OUTPUT BUFFER EXCEEDED"
H	Generate message prefix "DFH2506 OUTPUT EVENT REJ"
I	Generate message prefix "DFH2513 OUTPUT LENGTH ZERO"
J	Generate message prefix "DFH2514 NO OUTPUT AREA PROVIDED"
K	Generate message prefix "DFH2515 OUTPUT AREA EXCEEDED"
L	Generate message prefix "DFH2517 UNIT CHECK SNS="
M	Generate message prefix "DFH2519 UNIT EXCEPTION S.N.O."
N	Generate message suffix "AT TERM xxxx, TRANS yyyy, hh.mm.ss" or "ON LINE W/TERM, REL LINE zz, hh.mm.ss" or "ON LINE W/TERM, hh.mm.ss"
O	Generate message suffix "LINE, CNTRL, TERM xxxx, [REL LINE zz,] hh.mm.ss." or "LINE, CNTRL W/TERM xxxx, [REL LINE zz,] hh.mm.ss" or "LINE, TERM xxxx, [REL LINE zz,] hh.mm.ss" or "LINE W/TERM xxxx, [REL LINE zz,] hh.mm.ss" or "CNTRL, TERMxxxx, hh.mm.ss" or "CNTRL W/TERM xxxx, hh.mm.ss" or "TERM xxxx, hh.mm.ss"
P	Generate message prefix "DFH2508 UNAVAILABLE PRINTER"
Q	Write to terminal causing error
R	Write to destination "CSTL"
S	Write to destination "CSMT"
T	Obtain terminal main storage area (message build area)
U	Obtain transient data main storage
V	Generate message prefix "DFH2511 INVALID WRITE REQUEST"
W	Generate message infix "RETURN CODE xx"
X	Converts hexadecimal byte into two printable characters
Y	Generate message prefix "DFH2532 PRINT QUEUED"
Z	Generate message prefix "DFH2531 IC FAILURE X" where X can be "IOERROR", "TRNIDER", "TRMIDER", or "INVREQ"
AA	Not used
AB	Generate message "DFH2534 INVALID DESTINATION"
AC	Set terminal out of service
AD	Set line out of service
AE	Generate message prefix "DFH2504"
AF	Obtain terminal statistics
BA	Obtain line statistics
BB	Generate message prefix "DFH2516 UNIT CHECK SNS="
BC	Generate message prefix "DFH2518 UNIT EXCEPTION"
BD	Generate message infix ",S.N.O."
BE	Generate message prefix "DFH2520 NEGATIVE RESPONSE"
BF	Generate message prefix "DFH2521 UNDETERMINED UNIT ERROR"
CA	Generate message prefix "DFH2522 INTERCEPT REQUIRED"
CB	Generate message prefix "DFH2526 INTERV ON PRINTER" or "DFH2527 INTERV REQ" or "DFH2528 ERROR STATUS MSG XXXX RECEIVED"
CC	Not used
CD	Generate message prefix "DFH2523 INVALID COPY REQ"
CE	Generate message prefix "DFH2524 INVALID MSG BLOCK"
CF	Generate message prefix "DFH2525 INCMPLT MSG"
DA	Generate message prefix "DFH2510 TIME OUT"
DB	Generate message prefix "DFH2529 UNSOLICITED INPUT"
DC	Not used
DD	Not used
DE	Generate message prefix "DFH2530 INVALID READ REQUEST"
DF	Generate message prefix "DFH2509 INVALID DISC REQUEST"

Note: Messages have up to three parts; PREFIX, INFIX, and SUFFIX. Various routines assemble the parts, writing them to one of three destinations depending on the type of error.

Figure 201. TACP Message Routines

ERROR CODES	ROUTINE ORDER				
	81	8	1	2	7
	83	7			
	84	3	1/2	7	
	85	1	2	7	
	86	2	7		
	87	2	7		
	88	5	1	2	7
	89	6	1	2	7
	8A	8	1	2	7
	8B	1	2	7	
	8C	5	1	2	7
	8D	1	2	7	
	8E	1	2	7	
	8F	1	2	7	
	90	4	7		
	91	9	2	7	
	94	6	1	2	7
	95	6	1	2	7
	96	6	1	2	7
	97	6	1	2	7
	98	6	1	2	7
	99	5	1	2	7
	9A	7			
	9B	1	2	7	
	9C	6	1	2	7
	9D	6	1	2	7
	9E	2	7		
	A0	1	2	7	
	A1	2	7		

ROUTINE DESCRIPTION

1. Abend transaction
2. Link to DFHTEP
3. Put line in/out of service, as required
4. Unavailable printer processing
5. Put line or terminal out of service
6. I/O error test
7. Test line for next operation
8. Set line disconnect required
9. Unavailable printer interval control error.

Notes:

- The information given above is a generalization of TACP's default error handling.
- The left-hand column contains the error code.
- The right-hand column shows the routines used and the order in which they are used.

Figure 202. TACP Default Error Handling

Chapter 3.2. CICS Executable Modules

The following table shows:

1. Module names.
2. Entry points to the modules.
3. What calls the modules.
4. Brief descriptions of the modules.
5. What the modules call.
6. Which control blocks are referenced.

   Note: The names given in the **Control Blocks Referenced** column are the names of CICS copybooks or macro instructions that cause DSECTs to be included when the module is being assembled. The names of the DSECTs themselves may be obtained in the following way:

   a. For copybook and macro names that are preceded by an asterisk (*), refer to "Chapter 3.3. Control Block Copybook and Macro Names" on page 473.

   b. Copybook and macro names that are not preceded by an asterisk invoke a single DSECT of the same name.

7. What the modules exit to (where this column is blank, the module exits to the caller).

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHACEE	DFHACENA	DFHIRP DFHSIG1	The security block search module chains down the CVT and returns in registers 15 and 0: 0 or the user ID for the main ACEE.	None	None	
DFHACP	DFHACPNA	DFHDBP DFHPCP	The abnormal condition program writes a message to the terminal and to the CSMT destination if a transaction abends or cannot be started. Subject to tests on the type of terminal, DFHACP invokes DFHMGP to output the message. It calls DFHPEP, and depending on the result, may disable the transaction. For each error, there is an entry in a table that contains the number of the message to be written to the principal facility (terminal) and the number of the message to be written to CSMT. If, in either case, there is no message, then zero is entered. Listed below are the main subroutines of DFHACP: ABCSMTWT - Write to CSMT ACPCALMG - Use DFHMGP to output a message ACPCLPEP - Invoke DFHPEP ACPFENTY - Identify message for terminal TERMERR - Terminal error	DFHDCP DFHDIP DFHKCP DFHMGP DFHPCP DFHSCP DFHTRP DFHZARQ DFHZERH DFHZFRE DFHZRST	*DFHCSADS *DFHEIS *DFHLFM *DFHMGM *DFHPCTDS *DFHPPTDS *DFHTCA *DFHTCPCM DFHTCTFX DFHTCTLE *DFHTCTZE DFHTDOA DFHTIOA DFHZEPD	
DFHAKP	DFHAKPNA	DFHJCP	The activity keypoint program writes activity keypoints on to the system log. An activity keypoint is a collection of information describing the current state of the system. The presence of activity keypoints on the system log means emergency restart does not have to scan the whole log to ensure all required information has been seen. For standard-labeled journal tapes, DFHAKP writes records of all tapes constituting journals.	DFHJCP DFHKCP DFHPCP DFHTDP DFHVCP	*DFHCSADS DFHJCA *DFHSIT DFHTCA DFHVOLDS	
DFHALP	DFHALPNA	DFHCMP DFHCRQ DFHCRS DFHICP DFHTPQ DFHTPR DFHTPS DFHZATI DFHZISP DFHZNAC DFHZTSP	The terminal allocation program contains the logic to allocate TCTTE resources to requesting transactions. The request operates in a multiple exchange between the requesting transaction, and terminal control. A SCHEDULE request is communicated to terminal control as an ATI terminal control then responds with an AVAIL command. The requests are represented by AIDs (AID chain manipulations being performed by calls to DFHALP). For LU6.2, DFHALP issues a terminal control allocate mode name macro.	DFHKCP DFHSCP DFHTCP DFHTSP	*DFHAL *DFHCSADS *DFHDWEDS *DFHLFM *DFHPCTDS *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTZE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHAMP	DFHAMPNA	DFHEIP DFHSIT1	The allocation management program is invoked by the CEDA transaction. It analyzes commands and calls the definition file management program, DFHDMP, to process changes to records in the CSD. For the INSTALL command, DFHAMP also calls DFHPCP, DFHKCP, and DFHSPP. DFHPUP is called to convert data between address list format and the CSD record format.	DFHBTP DFHDMP DFHKCP DFHMGP DFHPCP DFHPPT DFHPUP DFHSCP DFHTRP DFHZCQ	DFHCSADS DFHMGM DFHTCA DFHTCTFX DFHTCTZE	
DFHAMTP	DFHAMTP	DFHAMP	DFHAMPT is called by the allocation management program to process requests for terminals and sessions during INSTALL and CHECK operations. When a definition is read, it is passed to the type object resolution module (DFHTOR).	DFHKCP DFHTOR DFHZCP	*DFHCSADS *DFHTCA DFHZCQPS	
DFHASV	DFHASVNA	DFHCSVC	DFHASV is the page fix/free/load interface. On entry to DFHASV, register 0 contains one of the following request codes: 0 Paging request 4 Termination request 5 Open files 6 Close files 12 Monitoring services - DFHCMP or DFHKCP 13 Monitoring services - DFHJCP 14 Monitoring services - DFHMCT 48 Journal tape volume serial dequeue 64 Authorize general purpose subtask TCB 80 Issue SDUMP		*DFHAFCD	Type 2 SVC

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHBFP	DFHBFPNA	User appl	In response to DFHBIF macro requests, the built-in functions program performs one of the following functions: • Table search • Phonetic conversion • Verification of a data field • Editing of a data field • Bit manipulation • Input formatting • Weighted retrieval. Listed below are the main subroutines of DFHBFP: BIFCHK - Verification of a data field TYPE = FVERIFY BIFDED - Editing of a data field TYPE = DEEDIT BIFINF - Input formatting TYPE = DEFLDNUM TYPE = INFORMAT BIFPHN - Phonetic conversion TYPE = PHONETIC BIFTSH - Table search TYPE = TSEARCH BIFTST - Bit manipulation TYPE = BITEST TYPE = BITFLIP TYPE = BITSETOFF TYPE = BITSETON BIFWGT - Retrieve selected records TYPE = WTRETGET BIFWPM - Establish selection criteria TYPE = WTRTPARM BIFWRL - Release weighted retrieval storage areas TYPE = WTRETREL BIFWRT - Initiate weighted retrieval TYPE = WTRETST	DFHFCP DFHSCP	*DFHCSADS *DFHSIT *DFHTCADS *DFHTCTTE DFHTIOA *DFHTRACE *DFHVSWA	
DFHBSIB3	DFHBSIB3	DFHTBSxx	DFHBSIB3 adds BMS 3270 support to a TCT table entry.		*DFHCSADS *DFHSIT	
DFHBSIZ3	DFHBSIZ3	DFHTBSxx	DFHBSIZ3 adds DFHZCP 3270 support to a TCT table entry.		DFHBSZGB *DFHCSADS *DFHSIT *DFHTCA DFHZCQDS	
DFHBSMIR	DFHBSMIR	DFHTBSxx	DFHBSMIR builds a TCT table entry for a session.		DFHBSZGB *DFHCSADS *DFHDWEDS *DFHSIT *DFHTCA *DFHTCTFX DFHZCQPS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHBSMPP	DFHBSMPP	DFHTBSxx	DFHBSMPP builds a TCT table entry for a pipeline pool entry.		DFHBSZGB *DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX DFHZCQPS	
DFHBSM61	DFHBSM61	DFHTBSxx	DFHBSM61 builds sessions for an LU6.2 mode group.	DFHZCQ	*DFHTCTFX DFHZCQPS	
DFHBSM62	DFHBSM62	DFHTBSxx	DFHBSM62 builds the mode-entry for an LU6.2 mode group.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSS	DFHBSS	DFHTBSxx	DFHBSS adds a new connection (system entry) to a CICS system.		*DFHCSADS *DFHDWEDS *DFHSIT *DFHTCA *DFHTCTFX	
DFHBSSA	DFHBSSA	DFHTBSxx	DFHBSSA initializes DFHKCP support in a new TCT system entry.			
DFHBSSF	DFHBSSF	DFHTBSxx	DFHBSSF initializes the stats counters in a new TCT system entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSS	DFHBSSS	DFHTBSxx	DFHBSSS builds DFHXSP support for a new TCT system entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZ	DFHBSSZ	DFHTBSxx	DFHBSSZ builds VTAM interface support for a new TCT system entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZB	DFHBSSZB	DFHTBSxx	DFHBSSZB adds a new batch interregion connection to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZG	DFHBSSZG	DFHTBSxx	DFHBSSZG adds a new advanced program-to-program communication (APPC) single-session connection to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZI	DFHBSSZI	DFHTBSxx	DFHBSSZI adds an indirect terminal control system table entry to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZL	DFHBSSZL	DFHTBSxx	DFHBSSZL adds a local terminal control system table entry to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSSZP	DFHBSSZP	DFHTBSxx	DFHBSSZP builds an advanced program-to-program communication (APPC) parallel-session to a CICS system.	DFHZCQ	*DFHCSADS *DFHSIT *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHBSSZR	DFHBSSZR	DFHTBSxx	DFHBSSZR builds an MRO session entry.		*DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX	
DFHBSSZS	DFHBSSZS	DFHTBSxx	DFHBSSZS builds an advanced program-to-program communication (APPC) session entry.		DFHBSZGB *DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX DFHZCQPS	
DFHBSSZ6	DFHBSSZ6	DFHTBSxx	DFHBSSZ6 builds an LU6.1 connection entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBST	DFHBST	DFHTBSxx	Performs TCTTE initialization common to terminals, pipeline pool entries, and sessions for IRC and ISC. DFHIRP		DFHTCTTE	
DFHBSTB	DFHBSTB	DFHTBSxx	DFHBSTB adds support for BMS to a new TCT terminal or session entry.		*DFHCSADS *DFHTCA	
DFHBSTB3	DFHBSTB3	DFHTBSxx	DFHBSTB3 adds partition support to a new TCT terminal or session entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTC	DFHBSTC	DFHTBSxx	DFHBSTC performs those operations that are executed after the installation of a terminal.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTD	DFHBSTD	DFHTBSxx	DFHBSTD adds data interchange program (DFHDIP) support for a new TCT table entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTE	DFHBSTE	DFHTBSxx	DFHBSTE adds EXEC diagnostic facility (EDF) support for a new TCT table entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTI	DFHBSTI	DFHTBSxx	DFHBSTI adds interval control program (DFHICP) support for a new TCT table entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTL	DFHBSTL	DFHTBSxx	DFHBSTL adds logical device code support to a new TCT terminal or session entry.		*DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX	
DFHBSTM	DFHBSTM	DFHTBSxx	DFHBSTM adds message generation program (DFHMGP) support for a new TCT table entry.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTP3	DFHBSTP3	DFHTBSxx	DFHBST adds 3270-copy support for a new TCT table entry.		*DFHCSADS *DFHSIT *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHBSTS	DFHBSTS	DFHTBSxx	DFHBSTS adds sign-on program (DFHSNP) support for a new TCT table entry.	DFHXSP	*DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX	
DFHBSTT	DFHBSTT	DFHTBSxx	DFHBSTT adds task control program (DFHKCP) support for a new TCT table entry.		*DFHAIDDS *DFHCSADS *DFHSIT *DFHTCA	
DFHBSTZ	DFHBSTZ	DFHTBSxx	DFHBSTZ builds a session or terminal resource.		DFHBSZGB *DFHCSADS *DFHDWEDS *DFHSIT *DFHTCA *DFHTCTFX *DFHZEPD DFHZCQPS	
DFHBSTZA	DFHBSTZA	DFHTBSxx	DFHBSTZA adds DFHZCP activity scan support to a new TCT terminal or session entry.	DFHALP	*DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX	
DFHBSTZB	DFHBSTZB	DFHTBSxx	DFHBSTZB appends or deletes a BIND image to a TCT terminal or session entry.	DFHZBAN	*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTZC	DFHBSTZC	DFHTBSxx	DFHBSTZC adds a single-session LU6.2 system as an advanced program-to-program (APPC) terminal.	DFHZCQ	*DFHCSADS *DFHTCA	
DFHBSTZH	DFHBSTZH	DFHTBSxx	DFHBSTZH adds an interregion (IRC) batch session to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTZO	DFHBSTZO	DFHTBSxx	DFHBSTZO adds an OS/VS console to a CICS system.		*DFHCSADS *DFHTCA *DFHTCTFX	
DFHBSTZR	DFHBSTZR	DFHTBSxx	DFHBSTZR adds an interregion (IRC) session to a CICS system.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTZS	DFHBSTZS	DFHTBSxx	DFHBSTZS adds an advanced program-to-program communication (APPC) session to the terminal control program.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSTZV	DFHBSTZV	DFHTBSxx	DFHBSTZV adds the parts of a terminal or session TCT table entry that are special to VTAM and IRC.		*DFHCSADS *DFHSIT *DFHTCA *DFHTCTFX *DFHZEPD	
DFHBSTZZ	DFHBSTZZ	DFHTBSxx	DFHBSTZZ adds a non-APPC session to the TCT. (APPC is advanced program-to-program communication.)		*DFHCSADS *DFHSIT *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHBSTZ1	DFHBSTZ1	DFHTBSxx	DFHBSTZ1 adds support for a remote terminal to a CICS system.	DFHZTSP	*DFHCSADS *DFHTCA *DFHTCTFX	
DFHBSTZ3	DFHBSTZ3	DFHTBSxx	DFHBSTZ3 adds a 3270 to the TCT.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSZZ	DFHBSZZ	DFHTBSxx	DFHBSZZ adds a terminal or session to the TCT.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSZZS	DFHBSZZS	DFHTBSxx	DFHBSZZS adds a new session to LU6.2 support.		*DFHCSADS *DFHSIT *DFHTCA	
DFHBSZZV	DFHBSZZV	DFHTBSxx	DFHBSZZV adds a VTAM terminal or session to the TCT.		*DFHCSADS *DFHSIT *DFHTCA	
DFHCAA70	DFHCAA70	BTAM	The 7770 channel end/abnormal end appendage handles start I/O completion codes, error conditions and normal completion for 7770 I/O operations.	None	DFHTCTLE	BTAM
DFHCAP	DFHCAPNA	DFHTCRP	DFHCAP processes command analysis for VTAM terminal definitions contained in a load module table DFHRDTxx.	DFHBEP DFHCUCA DFHCUCC DFHCUCD DFHD2EP	*DFHCSADS *DFHTCA	
DFHCCMF	DFHCCMNA	CCMF trans DFHCMON	The monitoring ATI program is used to periodically empty the data buffers associated with the various classes of monitoring. DFHCCMF is started at a frequency determined by the DFHMCT TYPE= RECORD,' FREQ=N statements. When DFHCCMF has completed its work, it scans the MCTXTOD list to determine the next time to start. This sequence is started and stopped by DFHCMON which is activated by the CSTT monitor command.	DFHCMP DFHICP DFHKCP DFHSCP	*DFHCSADS *DFHLFM DFHMCTDS DFHMCTRC	DFHCMON or DFHPCP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHCCP	DFHCCPNA		The catalog control program performs the following operations on a VSAM catalog: Open the catalog file Close the catalog file Establish a thread Release a thread Start a browse End a browse Get the next record Write a record to the catalog file Read a record from the catalog file Delete a record from the catalog file Purge records of a specified type.	DFHLFA	*DFHCSADS *DFHLFM *DFHTCA IFGACB IFGRPL	
DFHCMON	DFHCMONA	DFHSII1 DFHSTKC DFHSTP	The monitoring on line program changes the classes of active recording. DFHCMON is linked to by DFHSII1 to set up the environment as specified by the MONITOR SIT operand. The three classes are: ACCOUNT, PERFORM, and EXCEPTION. DFHCMON also outputs the class dictionary when the class is switched on. DFHCMON also produces a chain of pseudo-buffers which is anchored in the JCTTE and contains the dictionaries. DFHCMON is driven by DFHSTP if monitoring is active during CICS shutdown, and by DFHSTKC if the active classes of monitoring are changed by the CSTT command.	DFHCMP DFHICP DFHJCP DFHKCP DFHPCP DFHSCP	DFHCMPDR DFHCMPRC *DFHCSADS *DFHJCA *DFHJCTDS DFHJCTTE *DFHLFM DFHMCTDS DFHMCTRC DFHMCTSS *DFHTCA *DFHTCTFX DFHTCTTE DFHTIOA	DFHPCP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHCMP	DFHCMPNA	DFHEMP DFHTR (macros)	The monitoring program collects monitoring information and intercepts all calls to the trace program. DFHCMP decides if it wants to monitor the environment and calls DFHTRP if necessary.	DFHERM DFHICP DFHJCP DFHKCP DFHMSG DFHPCP DFHRMCAL DFHSCP DFHTRP DFHWTO	DFHAFCD DFHAICB DFHCMPDR *DFHCSADS DFHEDFDS DFHEIDDS DFHFCTDS DFHFCTSR DFHJCA DFHJCADS DFHJCSMF DFHJCTDS DFHJCTTE DFHMCT DFHMCTDS DFHMCTRC DFHMCTSS *DFHPAMDS *DFHPCTDS DFHSNT *DFHTCA DFHTCTFX DFHTCTTE DFHTDOA DFHTIOA DFHUEXIT	DFHEMP DFHTR (macros)
DFHCPY	DFHCPYNA	DFHPRK	The 3270 copy program (CSCY) causes data to be copied from screen to printer in a (VTAM) 3270 system. DFHCPY is invoked by DFHPRK (only if the 3270 has the copy feature) and issues a DFHTC TYPE=COPY macro to the printer. DFHCPY then initiates DFHRKB.	DFHICP DFHPCP DFHRKB DFHSCP DFHTCP	*DFHCSADS *DFHTCADS *DFHTCTZE DFHTIOA	
DFHCRC	DFHCRCNA	OS/VS	The interregion abnormal exit module is a CICS module that contains an (E) STAE exit to terminate interregion communication in abnormal conditions. DFHCRC issues a CLEAR request to the inter-region SVC.	DFHISP	*DFHAFCD *DFHIRSDS	Op sys
DFHCRNP	DFHCRNNA	DFHCRSP DFHKCP	DFHCRNP, the connection manager (transaction CSNC), controls IRC connections. It establishes, and breaks these connections and processes inbound requests to attach tasks (for example, mirror) to communicate with connected systems.	DFHFDP DFHIRP DFHISP DFHKCP DFHPCP DFHSCP DFHTDP DFHTRP	*DFHCRBDS *DFHCSADS DFHIRRDS *DFHIRSDS *DFHISCRQ *DFHMGM *DFHSEC *DFHTCA DFHTCTFX *DFHTCTZE DFHTIOA DFHZEPD	Suspends
DFHCRP	DFHCRPNA	DFHPCP	The multiregion operation relay program finds the system ID of a remote CICS system and calls DFHZTSP to ship the request to the other system.	DFHMGP DFHPCP DFHZTSP	*DFHCSADS *DFHISCRQ *DFHLFM *DFHMGM *DFHPCTDS *DFHTCA *DFHTCTZE DFHTIOA DFHZEPD	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHCRQ	DFHCRQNA	DFHPCP	The remote schedule page program is invoked periodically to delete requests to attach a transaction on a remotely owned terminal if those requests have been outstanding for more than the ATI purge delay interval.	DFHALP DFHICP DFHISP DFHMGP DFHPCP DFHSCP DFHTSP	DFHAL *DFHCSADS *DFHDCTDS *DFHLFM *DFHMGM *DFHTCA DFHZEPD	
DFHCRR	DFHCRRNA	DFHCRNP	The interregion session recovery program performs session recovery on behalf of primary or secondary IRC sessions.	DFHJCP DFHPCP DFHSCP DFHTCP	*DFHCSADS DFHIRRDS *DFHJCA *DFHMGM *DFHTCA *DFHTCTZE DFHTIOA	Detaches
DFHCRS	DFHCRSNA	DFHPCP	The remote scheduler program builds and ships AIDs for automatic transaction initiation when the terminal is in a remote address space. It receives requests to schedule an AID shipped to it from a remote address space.	DFHALP DFHMGP DFHPCP DFHSCP DFHZARQ DFHZLOC	*DFHAL *DFHCSADS DFHFMHDS *DFHLFM *DFHMGM *DFHPCTDS *DFHTCA *DFHTCTZE DFHTIOA	
DFHCRSP	DFHCRSNA	DFHEMB DFHSIJ1	The interregion communication startup module can be invoked either at system initialization or by CSMT request in order to make the CICS address space available for communication by other address spaces. DFHCRSP issues a logon request to the interregion communication SVC routine and attaches transaction CSNC (DFHCRNP).	DFHIRP DFHKCP DFHPCP DFHSCP DFHZDSP	*DFHCRBDS *DFHCSADS *DFHIRSDS *DFHSIT *DFHTCA DFHTCTFX *DFHTCTZE	
DFHCSA	DFHCSANA	Op sys	The module, DFHCSA, contains the common system area (CSA) and CSA optional features list, the dispatch control areas (DCAs), and task control areas (TCAs) for the task, and terminal control tasks, the timer interrupt exit routine, and the LIFO stack prolog routine. It also contains the runaway task abnormal termination routine, the queue control area (QCA), the paging control area, and, for HPO systems, the SRB interface control area.	DFHPCP	*DFHCSAD DFHDCADS DFHLFA *DFHPCTDS	Op sys
DFHCSDUP	DFHCUCNA	Op sys	The CSD utility program is an offline program that provides services for the CSD. The utility command processor (DFHCUCP) validates commands and invokes the appropriate routine to execute the requested function. DFHCSDUP calls DFHDMP to access the CSD.	DFHDMP DFHPUP		
DFHCSSC		Op sys	DFHCSSC, the security time-out program, is invoked by the transaction CSSC to scan TCTTEs at regular intervals to check for time-out after a terminal control free and to sign the terminal off, if required. The program is initiated by DFHXSP at sign-on, and subsequently reinitiates itself.	DFHICP DFHXSP	DFHLFM DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHCSVC	IGCxxx		This module is a type 2 SVC which passes control to the various required routines (shown in the calls column), dependent on the parameter passed to it. On a first request for a particular function it loads the required module and puts its address in the AFCB and then branches to that code. Further calls result in the address in the AFCB being branched to.	DFHASV DFHDEB7 DFHIRP DFHXSS	*DFHAFCD	Type 2 SVC
DFHCUCA	DFHCUCA	DFHTXRP	The resource definition online command analyzes interprets a VTAM resource definition in command form and produces a parameter list.			
DFHCUCB	DFHCUCB		The resource definition online command builder receives commands and transforms them to a format for use by the command processors.			
DFHCUCC	DFHCUCC	DFHTXRP	This program extracts a single entry from a loaded RDT table containing VTAM resource definitions.			
DFHCUCD	DFHCUCD	DFHTXRP	The resource definition online command default values program modifies the parameter list produced by DFHCUCA by inserting the default values.			
DFHCWTO	DFHCWTNA	CWTO	The console write to operator module is a CICS-provided transaction that allows an operator to send a message to the console operator. DFHCWTO issues SVC0 (EXCP) to pass the message to the operator's console.	DFHKCP DFHPCP DFHSCP DFHTCP	*DFHCSADS *DFHTCADS *DFHTCTLE *DFHTCTTE DFHTIOA	User
DFHDBP	DFHDBPNA	DFHPCP	The dynamic backout program is invoked following a transaction abend. It backs out changes made to recoverable resources by the abending transaction, thereby restoring their integrity, by using the dynamic log in main storage. DFHDBP can cause a transaction to restart. Listed below are the main subroutines of DFHDBP: DB0200 - Initialize and DWE scan DB0300 - Dynamic log scan DB0600 - Dynamic transaction backout logging	DFHACP DFHFCP DFHICP DFHKCP DFHPCP DFHRTY DFHSCP DFHTCP DFHTDP DFHTRP DFHTSP	*DFHCSADS DFHDBLDS DFHDBRDS *DFHDLP *DFHDWEDS *DFHFCTDS DFHFWADS *DFHPCTDS *DFHTCA *DFHTCTZE DFHTIOA *DFHUEXIT	DFHACP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHDCP	DFHDCPNA	API DFHFDP DFHPCP	The dump control program writes the short symptom string, and selected blocks and storage to an auxiliary data set processing by the dump utility program, DFHDUP. The areas which can be dumped include the PSW and registers, the CSA, TCA, TCTTE, TIOA, trace table, CICS tables and user transaction storage. DFHDCP also contains three subroutines that are used by the formatted dump program, DFHFDP, to write a formatted dump to the dump data set. These subroutines will, respectively, initiate a formatted dump, write a block and terminate a formatted dump.	DFHKCP DFHTRP	*DFHCSADS *DFHDCO DFHDCRDS DFHDCTDS *DFHLFM DFHLLADS *DFHMCBDS DFHOSPWA *DFHPCTDS *DFHPPTDS DFHPRADS DFHSAADS *DFHSCCOS *DFHSIT *DFHTCA *DFHTCTTE DFHTIOA *DFHTRACE	
DFHDEB70	DFHDEB70	BTAM	The 7770 DEB Processor (data event block).	None	None	BTAM
DFHDES	DFHDES	DFHZEV1 DFHZEV2 DFHZOPN	DFHDES performs data encryption and bind-time security.			
DFHDIP	DFHDIPNA	User appl DFHIIP DFHKCP DFHZSAP	The data interchange program acts as a function manager when transactions wish to communicate with batch devices using SNA support. DFHDIP builds and receives FMHs, which control the data set selection and function currently being performed by the batch device. Listed below are the main subroutines of DFHDIP: DESTCHEK - Destination change DIABORTE - Abort DICONRTE - Continue DIENDRTE - End DIINARTE - Transaction attach DIINPRTE - Input DINOTRTE - Note DIQUERTE - Query	DFHPCP DFHSCP DFHTCP DFHTRP DFHTSP	*DFHCSADS DFHDIBDS DFHFMHDS *DFHLFM *DFHTCA *DFHTCTZE DFHTIOA	
DFHDLBP	DFHDLBPA		DFHDLBP performs the backout of DL/I operations during emergency restart. It reads DL/I records relating to in-flight tasks from the restart data set, where they were written by DFHRUP, and sends them to the DL/I backout program in the DL/I system, scheduling an appropriate program specification block (PSB).	DFHLFA	*DFHCSADS *DFHDBO *DFHDLP *DFHFCTDS DFHFMIDS *DFHJCRDS *DFHSIT *DFHTCA DFSDBCO	Loops
DFHDLG	DFHDLGNA	IRLM	DFHDLG is used by the CICS-DL/I interface as an exit for IRLM to drive global commands. DFHDLG runs under a system transaction which is attached at initialization time. It is driven by a queue of work. Each work element is enqueued by the IRLM global command exit whenever another subsystem issued a global command for a data base.	DFHDLI DFHDLX DFHFDP DFHTDP	DFHCSADS DFHDLP DFHTCA IMS/VS	Loops

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHDLI	DFHDLINA	User appl DFHMTP DFHPCP DFHSPP DFHSTP DFHXFP	The CICS/MVS-DL/I interface program communicates directly with user written application programs, DL/I, and other CICS functions. The DL/I interface accepts requests for DL/I processing from application programs as well as CICS service modules. DFHDLI calls DFHDLX to perform certain internal functions. It also performs a security check on the resources used.	DFHDLX DFHFCP DFHICP DFHISP DFHJCP DFHKCP DFHPCP DFHSCP DFHSPP DFHXSP DL/I	*DFHCSADS *DFHDLP *DFHDWEDS *DFHFCTDS DFHFCTSR *DFHISCRQ *DFHJCADS *DFHJCTDS *DFHJCTTE DFHJCLDS *DFHOCTDS *DFHPPTDS DFHSEC *DFHTCA *DFHTCTZE *DFHTRACE	
DFHDLIAI	ASMTDLI CBLTDLI PLITDLI	User appl using DL/I CALL interface	This module is used by the CICS-DL/I interface. It is link-edited with the application program to provide communication between the application and the CICS-DL/I interface routine DFHDLI. Calls for DL/I to the entry point ASMTDLI, CBLTDLI, or PLITDLI are resolved by this processor.	DFHEIPDN	None	
DFHDLQ	DFHDLQNA	DFHSIP	DFHDLQ is the CICS-DL/I interface initialization program. DFHDLQ calls DFSRRC00 with appropriate parameters. On return from DFSRRC00, DFHDLQ allocates ISBs.	DFHSCP DL/I	*DFHCSADS *DFHDLP DFHFCTSR *DFHJCADS *DFHPCTDS *DFHSIT *DFHTCA *DFHTCTZE DFHTDOA	
DFHDLR	DFHDLRNA	IMS/VS	DFHDLR is the CICS service routine for DL/I. This routine performs task switching, buffer management, master terminal output of messages, logging and abending. DFHDLR also tracks IMS/VS EEQE records. These records have an IMS/VS log record identifier of 25, and are issued as a result of I/O error toleration (IOT) processing.	DFHJCP DFHKCP DFHPCP DFHTDP		
DFHDLRP	DFHDLRP		DFHDLRP is the DL/I restart program.	DFHLFA DFHDLBP	*DFHCSADS *DFHLFM *DFHSIT *DFHTCA	
DFHDLS	DFHDLSNA	IRLM	The status condition task is invoked when the local IRLM abends, or when communication with the remote IRLM (for interprocessor DL/I data sharing) fails. A system transaction is attached during initialization to service the requests.	DFHDBAUO DFHRDSHO	DFHCSADS DFHDLP DFHTCADS IMS/VS	Loops

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHDLX	DFHDLXNA	DFHDLI	DFHDLX processes internal DL/I requests, that is, requests issued by CICS modules to perform DL/I-related functions. The requests are passed to DFHDLX from DFHDLI in the form of standard parameter lists. The first parameter is a four-character string indicating the request type. The request types are supported in the following ways: REQUEST TYPE — ACTION OPEN, CLSE, DUMP, RCVR — Master terminal data-base commands. DBRC is informed of the command. If DBRC returns an error return code, then the command is rejected. Otherwise, if the command is GLOBAL, a DFSLM NOTIFY is issued to the IRLM to broadcast the command to data-sharing subsystems. Usage of the data base is quiesced where necessary, the data base is closed, and, for OPEN and DUMP calls, a change-authorization request is issued to DBRC by DFSDBAU0. *ACV — Inquires about the ACCESS value of a given data base (used by CEMT for DATASET INQUIRE). *BFD, *BFE — Backout failure during dynamic backout or emergency restart respectively. Stop all data bases updated by the failing logical unit of work and inform DBRC of the backout failure for each data base. In the case of dynamic backout failure, abend any tasks that have intent on the the failing data bases. Issue DFSLM NOTIFY to IRLM to broadcast the backout failure (INTERNAL STOP) to data sharing subsystems. *DLA — Informs DBRC of abnormal system log closure (during a CICS abend). Issued by the *TID routine. *DLC — Informs DBRC when the system log is closed. Called by DFHJCC. *DLI — Informs DBRC when an I/O error occurs on a system log. *DLO — Informs DBRC when the system log is opened or switched. Called by DFHJCO.	DFHDLI DFHDLX DFHFCP DFHJCP DFHKCP DFHPCP DFHSCP	*DFHCSADS *DFHDLP *DFHFCT *DFHJCADS *DFHJCTDS *DFHJCTTE DFHOCLDS *DFHPCTDS *DFHPPTDS *DFHTCA *DFHTCTZE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
			*DOC DBRC on-line command processor. Invoked by the CBRC master terminal transaction to pass DBRC on-line commands to DBRC for processing. Responses are passed to the caller.			
			*EOV Forces an end-of-volume on the system log (used by CEMT for an RCVR call).			
			*GTF Used by CEMT to obtain the status of the first data base in the DDIR (data base directory).			
			*GTN Used by CEMT to obtain the status of the next data base in the DDIR (data base directory).			
			*GTS Used by CEMT to obtain the status of a specified data base in the DDIR (data base directory).			
			*PIN DL/I initialization processing is performed at the end of CICS initialization. This includes issuing a DBRC signon-recovery-end call to allow authorizations to be freed after emergency restart (or cold start following a CICS abend). An IRLM PURGE is also issued to free locks held as a result of a CICS abend. DFSLM NOTIFY requests are issued to IRLM to broadcast that locks have been released (/RESUME ALLMSG) and to invalidate all buffers (for data base integrity). CSGX (global command task) and CSSX (status task), DFHDLG and DFHDLS respectively, are attached.			
			*QDL Shutdown stage 1 (before the system log is closed). Terminates the use of the data base monitor and unloads data bases by an UNLD call to IMS/VS.			
			*REC IRLM RECONNECT request from CEMT. Issued when the master terminal operator requests to reconnect the CICS subsystem to a new IRLM after the previous IRLM had failed. The routine issues an IDENTIFY to the IRLM, and, if successful, acquires the IRLM global command lock. The IRLM is then ready to be used.			
			*TDS Shutdown stage 2 (after the system log closed). Issues a SIGNOFF call to DBRC, a QUIT call to IRLM and awaits termination of the DBRC subtask.			

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
			*TID CICS system abend. Issues a *DLA (recursive) call to indicate that the system log is abnormally closed, an abnormal SIGNOFF call to DBRC and an abnormal QUIT call to IRLM.			
			*42R Builds an X'42'-type log record.			
DFHDLXA		DFHDLI	DFHDLXA is an extension of DFHDLX. It contains code to access the CICS catalog for CICS-DL/I calls, to support the IMS/VS I/O toleration processing of EEQE records, and to write status records for DDIR warm start. The list of control blocks referenced is in addition to those used by DFHDLX.		DFHCCRQ DFSTORSO	
	CATCALLS		The call types used by the common catalog routine are:			
			*CDL Delete a record from the catalog.			
			*CDL End a browse.			
			*CGN Get the next record from the catalog.			
			*CRD Read a record from the catalog.			
			*CSB Start a browse in the catalog for records of a specific restype. The browse begins at the first record of a given restype.			
			*CWR Write a record to the catalog. An existing record is overwritten, and a new record created.			
	EQE000		This routine reads the IMS/VS EEQEs that are saved in the catalog, and passes them to DFSTORSO, so that IMS/VS can re-instate them for this execution.			
	WDR000		This routine writes the status of some DDIR fields to the catalog.			

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHDMP		DFHAMP DFHCSDUP	The definition file management program handles physical changes to the CSD. Listed below are the main processes in DFHDMP: BUILDKWA (DM16) - Build key work area CONNECT (DM01) - CONNECT CREATSET (DM11) - Create SET DELETE (DM05) - DELETE DISCONN (DM02) - DISCONNECT ENDBRO (DM10) - End BROWSE ERASESET (DM12) - Delete SET GETNEXT (DM09) - Get next record LOCK (DM06) - LOCK QUERYSET (DM13) - QUERYSET READ (DM04) - Read CSD control records RELSEKWA (DM17) - Free key work area SETBRO (DM08) - Set browse UNLOCK (DM06) - UNLOCK WRITE (DM03) - WRITE	DFHFCP		
DFHDRP	DFHDRPNA	OS/VS	The dependent (batch) region control module is the first program executed by an IMS/VS batch job step. It opens the authorized program library DFHLIB which contains the remaining batch region controller modules (DFHDRPA-DFHDRPG). DFHDRP passes control to DFHDRPA.	None	None	DFHDRPA
DFHDRPA	DFHDRPAN	DFHDRP	The dependent (batch) region initialization module processes the user parameters in the JCL statements for an IMS/VS batch job. It establishes a connection with CICS and then passes control to DFHDRPB.	DFHDRPD DFHDRPE DFHDRPF	*DFHAFCD *DFHDRCA	DFHDRPB
DFHDRPB	DFHDRPBN	DFHDRPA	The application program control module initiates execution of the batch application program; when the batch program ends, DFHDRPB invokes DFHDRPC.	None	*DFHDRCA	DFHDRPC
DFHDRPC	DFHDRPCN	DFHDRPB	The batch region termination module sends a sync point request to CICS (using DFHDRPE) and invokes DFHDRPD to disconnect from CICS.	None	*DFHDRCA	DFHDRPD OS/VS
DFHDRPD	DFHDRPDN	DFHDRPC	The batch region cleanup and exits module includes the (E) STAE and SPIE exit routines and a cleanup routine (invoked when the batch job terminates to issue disconnect and logoff requests).	DFHIRP	*DFHDRCA *DFHIRSDS	DFHXFQ Op sys
DFHDRPE	DFHDRPEN	User appl DFHDRPC DFHDRPF	The DL/I request handling module consists of: • A program-request-handler routine which receives control when the application program issues a DL/I request. • A converse routine which sends requests to CICS by making switch requests to the SVC routine. The converse routine also handles responses.	DFHIRP DFHXFQ	*DFHDRCA *DFHIRSDS	DFHDRPF

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHDRPF	DFHDRPFN	DFHDRPA DFHDRPE	The SVC initialization module is invoked by DFHDRPA and issues a logon request and a connect request to the SVC routine; DFHDRPF also sends the DL/I scheduling request to CICS.	DFHIRP	*DFHDRCA *DFHIRSDS	
DFHDRPG	DFHDRPNA	DFHEIP	DFHDRPG is the EXEC interface processor for EXEC DLI commands for data base sharing. It receives the parameters of the command and from them builds a list that is appropriate to call DFHDRPE, the program request handler. On return from DFHDRPE, the status code in the PCB is examined. For some codes, an OS/VS abend is executed; the other codes are passed back to the application program.	DFHDRPE		
DFHDSB	DFHDSBNA	DFHPBP	The data stream build program produces the final device-dependent data stream for each page of BMS output. It is invoked only for processing data streams that are not in 3270 format. DFHDSB removes blanks from the ends of lines, converts logical new line characters into the device dependent equivalents (adding idle characters where necessary) and inserts horizontal and vertical tab characters if supported.	None	*DFHCSADS DFHFMHDS DFHMAPDS DFHOSPWA DFHSLDC *DFHTCA *DFHTCTTE DFHTIOA *DFHTTPDS	
DFHDUP	DFHDUPNA	Op sys	The dump utility program formats and prints output from dump management that has been generated during the execution of CICS. DFHDUP 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.	None	*DFHCSADS DFHDCRDS *DFHPCTDS *DFHSCCOS *DFHTCA	
DFHEAI	DFHEI1	User appl	The EXEC interface module is a processor that is link-edited with an assembler application program to provide communication with DFHEIP. The command-language translator turns each EXEC CICS command into a call statement. The external entry point invoked by the call is resolved to an entry point in this processor. The address of the entry point in DFHEIP (DFHEIPCN) is found through a chain of system and CICS control blocks.	DFHEIP	*DFHAFCD DFHAICB	
DFHEAI0	DFHEAI0	User appl	DFHEAI0 is a processor that is link-edited with an assembler application to provide communication with DFHEIP. The command-language translator generates calls to this processor to initialize and terminate the application program. The address of the entry point in DFHEIP (DFHEIPAN) is found using a chain of system and CICS control blocks.	DFHEIP	*DFHAFCD DFHAICB	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEAP	DFHEN001	Not applic	The assembler HLPI translator module performs the following functions: • Runs offline. • Takes on an input file. • Produces an output or listing file. • Gives a return code according to the highest severity of the message produced: 0 - no message 4 - warning 8 - error 12 - severe error 16 - translator failure. • It replaces CICS commands by invocations of the DFHEICAL macro, and inserts invocations of DFHEIENT, DFHEIRET, DFHEISTG, and DFHEIEND macros at appropriate places. Diagnostics resulting from errors in commands are inserted as comments in the output program and are not listed on the listing file.	None	None	Not applic
DFHEBF	DFHEBFNA	DFHEIP	The EXEC interface processor is for built-in function commands. For further information on this module refer to DFHEIP.	DFHBFP	*DFHEIPPL	DFHEIP
DFHEBU	DFHEBUNA	DFHETL DFHETC	The EXEC function management header (FMH) construction module is called by DFHETC when a SEND or CONVERSE command is being processed, and ATTACH function management headers have to be built and concatenated ahead of user data.	DFHSCP	*DFHEIPPL DFHFMHDS *DFHLFM *DFHTCTZE DFHTIOA	DFHETC
DFHECI	DFHEI1	User appl	The EXEC interface (COBOL) module is a processor similar to DFHEAI, except that it is used for COBOL application programs.	DFHEIP	*DFHAFCD DFHAICB *DFHCSADS	
DFHECID	DFHEIN01	DFHECIP	The command interpreter module analyzes CECI commands, and manages its displays. It uses the EXEC interface to invoke other CICS functions.	None	None	
DFHECIP	DFHEIN00	CECI trans	The command interpreter program performs preliminary validation and initialization for the CECI transaction, and links to DFHECID.	DFHECID	None	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHECP	DFHEN001	Not applic	The COBOL HLPI translator module performs the following functions: • Runs offline. • Takes on an input file. • Produces an output or listing file. • Gives a return code according to the highest severity of the message produced: 0 - no message 4 - warning 8 - error 12 - severe error 16 - translator failure. • DFHEIBLK and COMMAREA declarations are inserted in the LINKAGE section. • The EIB definition is inserted in the LINKAGE section. • The DIB definition (for DL/I HLPI) is inserted in the WORKING STORAGE section. • In the PROCEDURE DIVISION, the translator inserts a USING clause in the DIVISION statement, and replaces all CICS and DL/I commands by COBOL CALL statements.	None	None	Not applic
DFHEDAD	DFHESP01	DFHEDAP	The resource definition online (RDO) transactions module analyzes the commands, and manages the displays for CEDA, CEDB, and CEDC. It uses the EXEC interface.	None	None	
DFHEDAP	DFHESP00	CEDA, CEDB, CEDC trans	The resource definition online (RDO) transactions program performs preliminary validation and initialization for CEDA, and links to DFHEDAD.	DFHEDAD	None	
DFHEDC	DFHEDCNA	DFHEIP	The EXEC interface processor is for dump commands. For further information on this module refer to DFHEIP.	DFHDCP	*DFHEIPPL	DFHEIP
DFHEDFBR	DFHEDFBR	CEBR trans DFHEDFD	The temporary storage browse transaction browses, copies, or deletes entries in a temporary storage queue. It interprets commands and PF key actions.	DFHMCP	DFHAID DFHBMSCA *DFHEIB	
DFHEDFD	*	DFHEDFP	The EDF display program is invoked from DFHEDFP to analyze and display the current status of the user program. DFHEDFD stores control information on a temporary storage message queue and uses BMS to format the display screen. DFHEDFD interfaces with other CICS control programs using the EXEC interface.	DFHEDFBR	None	
	* Refer to microfiche					
DFHEDFM	Not applic	Not applic	The EDF map set contains BMS maps used by DFHEDFD to format the EDF display.	None	None	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEDFP	DFHEDFPN	DFHKCP	The EDF main program is the control program for EDF. DFHEDFP can be invoked in one of two ways: • It can be invoked directly from the EDF display terminal by entering the EDF transaction identification. • It can be invoked by hitting the user-defined PF key. DFHEDFP is also attached by DFHEDFX as the main program of the EDF task.	DFHEDFD DFHICP DFHKCP DFHPCP DFHSCP DFHTCP DFHTSP	None	
DFHEDFR	DFHEDFRN	Not applic	The EDF response table contains a description of the exception responses for each EXEC command and the abend codes associated with error responses. DFHEDFR is used by DFHEDFD to interpret the responses obtained from an EXEC command.	None	None	
DFHEDFX	DFHEDFXN	DFHACP DFHEIP DFHPCP	The EDF task switch program is invoked from DFHEIP, DFHPCP, or DFHACP when a program is running in debug mode. DFHEDFX suspends the user task and attaches the debugging task passing it information about the user task in the TWA of the debugging task.	DFHFCP DFHKCP DFHPCP DFHTCP	None	
DFHEDI	DFHEDINA	DFHEIP	The EXEC interface processor is for data interchange commands. For further information on this module refer to DFHEIP.	DFHDIP	*DFHEIPPL *DFHTCTTE DFHTIOA	DFHEIP
DFHEDP	DFHEDPNA	DFHEIP	DFHEDP converts command-level DL/I statements into a called parameter list acceptable to DL/I. In addition, it provides 31 bit application support by moving segment I/O areas above and below the 16-megabyte line as required.	DFHDLI DFHPCP DFHSCP	*DBPCB DFHCSAD DFHRSBDS DFHTCA DFHTCADY DFHUEPAR	
DFHEEI	DFHEEINA	DFHEIP	The EXEC interface processor is for ADDRESS, ASSIGN, PUSH, POP, and HANDLE commands, and performs the following functions: • It obtains information from CICS control blocks. • It establishes the action to be taken by DFHEIP when EXEC error conditions occur. For further information on this module refer to DFHEIP.	None	*DFHEIPPL *DFHDCTDS DFHOSPWA DFHPCTDS *DFHTACB *DFHTCTFX DFHTCTTE	DFHEIP
DFHEEX	DFHEEXNA	DFHETC	The EXEC function management header (FMH) extraction module is called by DFHETC when a RECEIVE or CONVERSE command is being processed, and when data has to be extracted from ATTACH function management headers.	None	*DFHEIPPL DFHFMHDS *DFHLFM *DFHTCTZE DFHTIOA	DFHETC

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEFC	DFHEFCNA	DFHEIP	The EXEC interface processor is for file control commands. For further information on this module refer to DFHEIP.	DFHFCP	*DFHEIPPL *DFHFCTDS DFHFIOA DFHFWADS *DFHSEC *DFHVSWA	DFHEIP
DFHEGL	DFHEGLNA	DFHEIP	DFHEGL processes LU6.2 requests. If the request is valid, a DFHLUC macro is issued.	DFHAMP DFHLFO DFHZARL	DFHLUCDS DFHTIOA DFHZEPD	
DFHEIC	DFHEICNA	DFHEIP	The EXEC interface processor is for interval control commands. For further information on this module refer to DFHEIP.	DFHICP	*DFHEIPPL *DFHPCTDS *DFHSEC *DFHTSHD	DFHEIP
DFHEIDTI	DFHEIDNA	DFHEIP	The EXEC interface processor for ask-time and format-time. DFHEIDTI updates the time and date fields in the EIB, certain time fields in the CSA, and returns the current time, and/or date, to the application.		*DFHCSADS *DFHTCA	DFHEIP
DFHEIP	DFHEIPNA	DFHPCP	The EXEC interface program consists of a nucleus and one EXEC interface processor for each CICS management module. DFHEIP is first invoked by DFHPCP when any command-level program (including the mirror program, DFHMIR) is loaded. DFHEIP initializes the EXEC interface structure (EIS) and then invokes the application program. Each EXEC CICS command invokes DFHEIP (nucleus) which in turn invokes the appropriate interface processor. The processor: 1. Passes application parameters to CICS control blocks. 2. Invokes the appropriate management module. 3. Passes data back to the application through passed parameters. DFHEIP also returns information to the application program through EIB (within EIS).	DFHAMP DFHEBF DFHEDC DFHEDFX DFHEDI DFHEEI DFHEFC DFHEGL DFHEIC DFHEJC DFHEKC DFHELR DFHEMS DFHEPC DFHERM DFHESC DFHESP DFHETC DFHETD DFHETL DFHETR DFHETS	*DFHCSADS *DFHDWCMN *DFHEIMDS *DFHEIS *DFHJCADS DFHOCLDS *DFHPCTDS *DFHPPTDS *DFHSEC *DFHTCA *DFHTCTTE *DFHTRACE *DFHUEXIT	DFHPCP
DFHEIQDS	DFHEIQDS	DFHEIP	The EXEC interface processor for inquire and set commands for data sets.		*DFHCSADS DFHDSNDS DFHFCTDS *DFHTCA	DFHEIP
DFHEIQSA	DFHEIQSA	DFHEIP	The EXEC interface processor for inquire and set commands for system attributes.		*DFHCSADS *DFHEIB *DFHEIS *DFHTCA	DFHEIP
DFHEIQSC	DFHEIQSC	DFHEIP	The EXEC interface processor for inquire and set commands for connections.		*DFHCSADS *DFHSNT *DFHTCA DFHTCTSK *DFHTCTTE	DFHEIP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEIQSM	DFHEIQSM	DFHEIP	The EXEC interface processor for inquire and set commands for modenames.		*DFHCSADS *DFHEIB *DFHEIS *DFHTCA *DFHTCTTE	DFHEIP
DFHEIQSP	DFHEIQSP	DFHEIP	The EXEC interface processor for inquire and set commands for programs.		*DFHCSADS *DFHPPTDS *DFHTCA	DFHEIP
DFHEIQST	DFHEIQST	DFHEIP	The EXEC interface processor for inquire and set commands for terminals.		*DFHCSADS *DFHSNT *DFHTCA DFHTCTSK *DFHTCTTE	DFHEIP
DFHEIQSX	DFHEIQSX	DFHEIP	The EXEC interface processor for inquire and set commands for transactions.		*DFHCSADS *DFHPCTDS *DFHTCA	DFHEIP
DFHEITAB			DFHEITAB is the translator table for EXEC CICS commands.			
DFHEJC	DFHEJCNA	DFHEIP	The EXEC interface processor is for journal control commands. For further information on this module refer to DFHEIP.	DFHTCP	*DFHEIPPL *DFHJCADS *DFHJCTDS *DFHJCTTE *DFHSEC	DFHEIP
DFHEKC	DFHEKCNA	DFHEIP	The EXEC interface processor is for task control commands. For further information on this module refer to DFHEIP.	DFHKCP	*DFHEIPPL	DFHEIP
DFHELR	DFHELRNA	DFHEIP	The local/remote decision routine (invoked by DFHEIP) determines if the required resource is local or remote. (A resource is remote if the SYSID option is specified or if the entry in the associated FCT, DCT, TST, or PCT shows the resource to be remote.) If local, control is returned to DFHEIP; if remote, DFHELR locates (or creates) a transformer storage area XFSTG and issues a DFHIS TYPE=CONVERSE macro. When the reply is received from the remote system, DFHELR returns control to DFHEIP.	DFHFCP DFHISP DFHKCP DFHSCP DFHTDP DFHTSP	*DFHDCTDS *DFHEIPPL *DFHFCTDS *DFHISCRQ *DFHLFM *DFHPCTDS *DFHSEC *DFHTCTTE DFHTSTDS	DFHEIP
DFHEMA	DFHEMANA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • Task-values • TCLASS • BATCH • RESET	DFHICP DFHKCP	*DFHEIMDS *DFHEIPPL *DFHTCTTE	DFHEIP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEMB	DFHEMBNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • VTAM • DUMP • IRC • IRBATCH • AUXTRACE • TRACE • PERFORM • PITRACE • SNAP • SHUTDOWN	DFHICP DFHKCP DFHPCP DFHTCP DFHTRP	*DFHCRBDS *DFHDCO *DFHEIMDS *DFHEIPPL *DFHIRSDS DFHTCTFX DFHTCTLE DFHTCTTE DFHZEPD	DFHEIP DFHSTP
DFHEMC	DFHEMCNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • TERMINAL • NETNAME	DFHKCP DFHPCP DFHTCP	*DFHEIMDS *DFHEIPPL *DFHPCTDS *DFHTCTZE DFHZEPD	DFHEIP
DFHEMD	DFHEMDNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • SYSTEM • MODENAME	DFHKCP DFHTCP	*DFHEIMDS *DFHEIPPL *DFHTCTZE *DFHZEPD	DFHEIP
DFHEME	DFHEMENA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • TRANSACTION • TASK	DFHKCP DFHPCP DFHTCP	DFHDCADS *DFHDCTDS *DFHEIMDS *DFHEIPPL *DFHPCTDS *DFHPPTDS *DFHTCTTE *DFHZEPD	DFHEIP
DFHEMF	DFHEMFNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • PROGRAM • QUEUE	DFHKCP DFHPCP	*DFHDCTDS *DFHEIMDS *DFHEIPPL DFHOCLDS DFHOCODS *DFHPPTDS	DFHEIP
DFHEMG	DFHEMGNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • CONTROL • LINE	DFHKCP DFHPCP DFHTCP	*DFHEIMDS *DFHEIPPL DFHTCTLE *DFHTCTTE DFHZEPD	DFHEIP
DFHEMH	DFHEMHNA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • VOLUME • JOURNAL	DFHJCP DFHVCP	*DFHEIMDS *DFHEIPPL DFHJCTTE DFHVOLDS	DFHEIP
DFHEMI	DFHEMINA	DFHEIP	The enhanced master terminal routine which is invoked by the following keywords: • DATASET • DLIDATABASE • RECONNECT	DFHDLI DFHFCP	*DFHEIMDS *DFHEIPPL *DFHFCTDS DFHOCLDS	DFHEIP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHEMS	DFHEMSNA	DFHEIP	The EXEC interface processor is for BMS commands. For further information on this module refer to DFHEIP.	DFHMCP	*DFHEIPPL DFHOSPWA *DFHTCTTE DFHTIOA	DFHEIP
DFHEMTA	DFHEMT00	User appl	The master terminal programmed interface program is a special version of DFHEMTP that a user application can link to for for master terminal services.	DFHEMTD	None	
DFHEMTD	DFHEMT01	DFHEMTA DFHEMTP DFHEOTP DFHESTP	The master terminal module analyzes the commands, and manages displays for CEMT, CEOT, and CEST transactions. It uses the EXEC interface.	None	None	
DFHEMTP	DFHEMT00	CEMT trans	The master terminal program performs preliminary validation and initialization for the CEMT transaction, and links to DFHEMTD.	DFHEMTD	None	
DFHEOTP	DFHEMT00	CEOT trans	The master terminal program performs preliminary validation and initialization for the CEOT transaction, and links to DFHEMTD.	DFHEMTD	None	
DFHEPC	DFHEPCNA	DFHEIP	The EXEC interface processor is for program control commands. For further information on this module refer to DFHEIP.	DFHPCP	*DFHEIPPL *DFHPPTDS *DFHSEC *DFHTCTTE	DFHEIP
DFHEPI	DFHEI1	User appl	The EXEC interface (PL/I) module is a processor similar to DFHEAI, except that it is used for PL/I programs.	DFHEIP	*DFHAFCD DFHAICB	
DFHEPP	DFHEN001	Not applic	The PL/I HLPI translator module performs the following functions: • Runs offline. • Takes on an input file. • Produces an output or listing file. • Gives a return code according to the highest severity of the message produced: 0 - no message 4 - warning 8 - error 12 - severe error 16 - translator failure	None	None	Not applic

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
			• If the input program is a MAIN procedure the PL/I HLPI translator inserts DFHEIPTR as the first parameter on the PROCEDURE statement to address the EIB. The translator also inserts declarations of the EIB and certain temporary variables. • Any CICS or DL/I commands in the input program are replaced by CALL statements in the output program. • Any errors in commands are diagnosed in messages on the translator listing file.			
DFHEPS	DFHEPSNA	DFHEIP	DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.	DFHPSP	*DFHCSADS DFHEIB *DFHEIS DFHPSGPS *DFHTCA	DFHEIP
DFHERM	DFHERMNA	DFHEIP	DFHERM is called by DFHEIP on behalf of the other components of CICS to manage the connection between CICS and non-CICS products.	DFHEIP routines Task-related user exits	DFHAFCD DFHCSA DFHEIB DFHEPB DFHTCA DFHTCTTE DFHTIEDS *DFHUEXIT	
DFHESC	DFHESCNA	DFHEIP	The EXEC interface processor is for storage control commands. For further information on this module refer to DFHEIP.	DFHSCP	*DFHEIPPL	DFHEIP
DFHESP	DFHESPNA	DFHEIP	The EXEC interface processor is for sync point commands. For further information on this module refer to DFHEIP.	DFHSPP	*DFHEIPPL	DFHEIP
DFHESTP	DFHEMTOO	CEST trans	The master terminal program performs preliminary validation and initialization for the CEST transaction, and links to DFHEMTD.	DFHEMTD	None	
DFHETC	DFHETCNA	DFHEIP	The EXEC interface processor is for terminal control commands. It synchronizes I/O for terminal control commands. The equivalent synchronization for BMS and data interchange commands is performed in DFHEIP.	DFHEBU DFHETL DFHTCP	*DFHEIPPL DFHFMHDS *DFHPCTDS *DFHTCTZE DFHTIOA	DFHEIP
DFHETD	DFHETDNA	DFHEIP	The EXEC interface processor is for transient data commands. For further information on this module refer to DFHEIP.	DFHTDP	*DFHDCTDS *DFHEIPPL *DFHSEC DFHTDIA DFHTDOA	DFHEIP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHETL	DFHETLNA	DFHETC	The EXEC interface processor is for LU6.2 commands.	DFHAMP DFHLFO DFHSCP DFHZARL DFHZARM DFHZARQ	DFHLUCDS DFHTIOA DFHZEPD	
DFHETR	DFHETRNA	DFHEIP	The EXEC interface processor is for trace commands. For further information on this module refer to DFHEIP.	DFHTRP	*DFHEIPPL	DFHEIP
DFHETS	DFHETSNA	DFHEIP	The EXEC interface processor is for temporary storage commands. For further information on this module refer to DFHEIP.	DFHTSP	*DFHEIPPL *DFHSEC *DFHTSHD DFHTSTDS	DFHEIP
DFHEXI	DFHEXINA	DFHZARQ	The exceptional input program is linked to from DFHZCP when unexpected input is received from a VTAM 3270 terminal that has a task attached. DFHEXI checks whether the input is the result of a 3270 print function key being pressed; if so, DFHEXI issues a DFHTC TYPE=PRINT macro, and then unlocks the keyboard; in any case, DFHEXI then passes control back to DFHZCP.	DFHPCP DFHSCP DFHTCP	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTZE DFHTIOA	DFHZCP
DFHFCBP	DFHFCBNA		DFHFCBP performs file backout during emergency restart. File backout records for in-flight tasks are read from the restart data set, where they were written by DFHRUP. The records are reinstalled to CICS files using calls to file control.	DFHLFA	*DFHCSADS *DFHFBO *DFHFCTDS DFHFMIDS *DFHFWADS *DFHJCRDS *DFHLFM DFHRCRQ *DFHSIT *DFHTCA *DFHUEXIT	
DFHFCC	DFHFCCNA		DFHFCC is a file control program that is link-edited into DFHFCP. DFHFCC handles requests to locate, or browse, entries in the file control table (FCT).	DFHLFA	*DFHCSADS *DFHFCTDS DFHFCWS *DFHLFM *DFHTCA DFHTMRQ	
DFHFCD	DFHFCDNA	DFHFCP	DFHFCD is a file control program that is link-edited into DFHFCP. DFHFCD handles BDAM file control requests except for OPEN and CLOSE.	DFHKCP DFHPCP	*DFHCSADS *DFHDWEDS DFHFBWA *DFHFCTDS DFHFCWS DFHFIOA DFHFWADS *DFHJCA *DFHLFM *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHFCJ	DFHFCJNA	DFHFCP DFHFCD	DFHFCJ is a file control program that is link-edited into DFHFCP. DFHFCJ handles journaling requests.	DFHLFA	*DFHCSADS DFHDSNDS *DFHFCTDS DFHFCWS DFHFMIDS *DFHJCA *DFHLFM *DFHTCA	
DFHFCL	DFHFCL	DFHFCN	DFHFCL is a file control program that is link-edited into DFHFCS. DFHFCL builds and deletes VSAM LSR pools. It is called by DFHFCN with a parameter list which specifies the pool number (1-8), and the action to be taken (build or delete). See also DFHSIE1.		*DFHCSADS *DFHDSNDS DFHFCSBK DFHFCSDS *DFHFCTDS *DFHFCTSR *DFHTCA	
DFHFCN	DFHFCN		DFHFCN is a file control program that is link-edited into DFHFCS. DFHFCN opens and closes files. If a file has has not been allocated, DFHFCN allocates it, and frees it on closure.	DFHFCL DFHFCS DFHLFA DFHSKP DFHTDP DFHTMP	*DFHCSADS *DFHDSNDS DFHDSSDS DFHFCSDS *DFHFCTDS *DFHFCTSR *DFHLFM *DFHTACB *DFHTCA *DFHTDOA IFGACB	
DFHFCP	DFHFCPNA	User appl	The file control program provides file services for application programs and for other CICS modules. These services include: • Acquisition and releasing of the file storage by storage control • Communication with files defined in the file control table • Logging of changes to these files by DFHFCJ and journal control. DFHFCP handles requests that access data in VSAM files. Requests that access BDAM data are passed to DFHFCD. Requests that change the file state (for example, OPEN, ENABLE) are passed to DFHFCS, regardless of the access method. DFHFCS is invoked using the DFHPC TYPE=LINK macro.	DFHFCC DFHFCD DFHFCJ DFHFCV DFHKCP DFHPCP DFHSCP DFHTRP	*DFHCSADS DFHDSNDS DFHDSSDS *DFHDWEDS DFHFCSBK *DFHFCTDS DFHFCTSR DFHFCWS DFHFIOA DFHFWADS DFHHTADS *DFHLFM *DFHTCA DFHVSWA	
DFHFCRP	DFHFCRP		The file control restart program restores the FCT and invokes DFHFCBP to perform file backout.	DFHCCP DFHLFA DFHPCP DFHTMP	*DFHCSADS DFHDSSDS DFHFCSDS *DFHFCTDS *DFHLFM *DFHSIT *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHFCS	DFHFCSNA		DFHFCS changes the state of a file. It invokes DFHFCN to open, or close, files.	DFHFCN	DFHCCRQ *DFHCSADS DFHDSNDS DFHDSSDS *DFHDWEDS *DFHFCTDS *DFHTCA *DFHUEXIT *DFHVSWA	
DFHFCU	DFHFCUNA	CSFU trans	DFHFCU issues an OPEN for files specified in the file control table (FCT). This program examines the FCT, and issues the DFHFC macro to open all specified files.	DFHFCP	*DFHCSADS *DFHFCTDS *DFHOCLDS *DFHTCA	
DFHFCV	DFHFCVNA		DFHFCV is a file control program that is link-edited into DFHFCP. It handles requests to VSAM, and some of the responses.		*DFHCSADS *DFHFCTDS *DFHFCTSR DFHFCNS *DFHTACB *DFHTCA DFHVSCA *DFHUEXIT *DFHVSWA IFGACB	
DFHFCX	DFHFCXNA UPADEXIT		DFHFCX is the file control program that handles the UPAD exit. The DFHFCXNA entry is used on the first call for initialization and UPADEXIT is used to enter the UPAD routine.		*DFHAFCD *DFHCSADS *DFHFCTDS *DFHPCTDS *DFHTCA *DFHVSCA *DFHVSWA IFGACB	
DFHFDB	DFHFDBNA	DFHFDP	The formatted dump module B performs the bulk of the work involved in producing a formatted dump. It is written in the form of an interpreter. It formats the control blocks, detects errors, and issues error messages.	None	None	DFHFDP
DFHFDC	Not applic	Not applic	The formatted dump module C contains no executable code. It consists of the tables which are interpreted by DFHFDB to control the blocks printed and the fields which are highlighted.	Not applic	All CICS control blocks	Not applic

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHFDP	DFHFDPNA	API DFHEMT DFHMTPA DFHSCP DFHSCR DFHSIP DFHSRP DFHSTP	The formatted dump program produces a formatted dump of the CICS address space: • In response to a CEMT or CSMT command. • If a storage violation occurs and a dump is requested. • If shutdown with dump is requested. • If a transaction abends with a code of ASRA or ASRB and the PCT requests a dump. • If CICS abends. The first line of the dump is the short symptom string. In all cases the dump is written to the CICS dump data set using the subroutines provided by DFHDCP. The dump may then be formatted offline using DFHDUP.	DFHDCP DFHFDB	All CICS control blocks	
DFHFEP	DFHFEPNA	Not applic	The FE terminal test program is invoked by the transaction CSFE. It can be used to send a complete character set to a terminal or to echo input or to turn tracing on or off. This program is an application program and does not exit to any other CICS modules. However it does use CICS facilities.	DFHKCP DFHPCP DFHSCP DFHTCP	*DFHCSADS *DFHLFM *DFHMGM *DFHPCTDS *DFHTCA *DFHTCPCM DFHTCTFX DFHTCTLE *DFHTCTZE DFHTIOA	Not applic
DFHFTAP	DFHFTANA	Op sys	The format-tape program is used to preformat journal tapes so that DFHTEOF can write an end-of-file mark at the end of the journaled data.	None	None	Op sys
DFHGMM	DFHGMMNA	DFHKCP	The good morning program is invoked by the system transaction CSGM to write a "good morning" message to VTAM logical units when a satisfactory OPNDST has occurred (and provided that the message has been requested in the TCT TYPE=TERMINAL entry).	DFHSCP DFHTCP	*DFHCSADS *DFHLFM *DFHTCA DFHTCTFX *DFHTCTZE DFHTIOA	
DFHHPSVC	IGCnnn	DFHKCP (Via an SVC call	This is a type 6 SVC module used only on MVS. Its sole purpose is to cause MVS to dispatch an SRB. DFHHPSVC provides part of the CICS high performance option (HPO) code, and is invoked only if HPO is in use. In the entry point, nnn is the number of the SVC.	None	*DFHAFCD	MVS

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHICP	DFHICPNA	DFHEIC applic DFHKCP DFHLUP	The interval control program is used for time management and has two main functions: • To service DFHIC macros under control of a requesting task's TCA. • To detect the expiration of time-dependent events for the task dispatcher, as defined in ICEs. Listed below are the main subroutines of DFHICP: ICCANCLN - Cancel a time-ordered request ICEXPANL - Time expiration analysis ICGTIMEN - Current time of day ICGTTTDM - Data retrieval ICICECRN - Build basic ICE ICPCTSN - Task initiation ICPOSTN - Signal expiration of a specified time ICRESETN - Time of day clock reset support ICSCHEDN - ICE schedule ICWAITN - Delay processing of a task	DFHKCP DFHSCP DFHTCP DFHTSP	*DFHAIDDS *DFHAL DFHCSADS *DFHDWEDS *DFHICEDS *DFHLFM *DFHPCTDS *DFHTCA *DFHTCTZE *DFHUEXIT	
DFHIIP	DFHIIPNA	DFHMCP	The non-3270 input mapping program performs all BMS input mapping functions for all devices except the 3270. On exit from the module, the input data has been mapped into a newly acquired TIOA that is returned to the application program and is then addressable using BMS DSECTs in the application. Listed below are the main subsections of DFHIIP: IIMID - GETMAIN TIOA to return to user and map page buffer into it using the specified map. IIREAD - Read input data issues DFHTC or DFHDI requests to obtain data from the terminal. IISCAN - Scan data stream for device dependent control characters and create page buffer.	DFHDIP DFHMCP DFHPCP DFHSCP DFHTCP	*DFHCSADS DFHMAPDS DFHOSPWA *DFHTCA *DFHTCTZE DFHTIOA *DFHTTPDS	
DFHIRP	DFHIRPNA	DFHCRC DFHCRNP DFHCRSP DFHDRPD DFHDRPE DFHDRPF DFHSRP DFHSTP DFHZCX	The interregion communication program is used to pass data from one region to another in the same CEC. The programs executing in the regions will usually be CICS programs, but DFHIRP does not assume this.		DFHAFCD DFHIRPD DFHIRSDS DFHSABDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHISP	DFHISPNA	DFHDLI DFHELR	The intersystem communication program is invoked when a request to access resource has to be shipped to a remote system (through ISC or MRO). The requests passed to DFHISP are: • File control • Interval control • Temporary storage • Transient data • DL/I DFHISP controls the acquisition, use, and freeing of a session to the remote system and invokes DFHXFP or DFHXFX to process requests and replies. A user exit is provided in DFHISP to support the local queueing of EXEC START commands when the link to the remote system is out of service. If a request is queued, DFHISP passes the transaction CMPX, which is started when the link is brought in to service.	DFHISP DFHKCP DFHSCP DFHTCP DFHXFP DFHXFX DFHZISP	*DFHISCRQ *DFHLFM DFHTSIOA *DFHTCPSV *DFHUEXIT	
DFHJCBSP	DFHJCBNA	DFHSIP	The journal control bootstrap program, if called initially, waits on completion of the open/close subtask and then checks good completion. If called for a journal task, DFHJCBSP acquires a journal buffer, sets up the buffer control entries in the JCT, and then becomes a journal task which continues to run until terminated by shutdown.	DFHICP DFHJCP DFHKCP DFHPCP DFHSCP	*DFHCSADS DFHDCADS *DFHJCA DFHJCOCL *DFHJCTTE DFHSAADS *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCC	DFHJCCNA	DFHJCEOV DFHJCIOE DFHJCO	The journal control close program: • Checks status of journal and sets a close option (rewind, unload, or leave). • Enqueues to single-thread a subtask. • Passes a request to a subtask (DFHJCOCP) using a POST. • Waits for a request to complete. • If journaling to disk, calls DFHXJCC after making the request to the subtask. • For standard-labeled output volumes, passes a trailer label. • Records the status and time if a standard-labeled volume is being closed. • Issues a CICS wait for the requesting task. • Calls DFHVCP to build or read a user label, and inform volume management of the new status (closed or defective) of the volume. • Dequeues, to release single threading. • If system log, records event in RSD catalog. • Puts the journal into exclusive control mode. • For systems with data sharing, calls DFHDLI to inform DBRC of the closure. • Returns control to DFHJCP.	DFHCCP DFHJCOCP DFHKCP DFHPCP DFHVCP DFHXJCC *DFHVOLDS	*DFHCSADS *DFHJCA DFHJCOCL *DFHJCTTE *DFHLFM *DFHTCADS	
DFHJCEOV	DFHJCENA	DFHJCP	The journal control output end-of-volume program when called by DFHJCP (at end-of-volume), closes the currently open output volume and opens a new output volume by issuing DFHJC macros. If this is not successful, either CICS or the journal task is abended, and message DFH4512 is written to the operator. For standard-labeled journal tapes, the DFHVC macro is used to select the volume to open and inform the operator of the succeeding volume. For two drives, the new volume is opened before the old volume is closed.	DFHJCP DFHJCSDJ DFHKCP DFHPCP DFHSCP DFHTDP DFHVCP DFHWTO	*DFHCSADS *DFHJCA *DFHJCTDS *DFHJCTTE *DFHLFM *DFHTCA *DFHTCTFX *DFHTDOA DFHVOLDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCI	DFHJCINA	DFHJCP	The journal control input program is called from DFHJCP to process any input requests, specifically: • GETF (or GETB) retrieves the next (or previous) logical record from the specified journal. • NOTE retrieves current journal positioning data. • POINT repositions the journal. For standard-labeled tapes, DFHJCI calls DFHVCP to fill in NOTE blocks and to record any failure on input that occurs.	DFHKCP DFHPCP DFHSCP DFHVCP DFHWTO	*DFHCSADS *DFHJCA DFHJCICA *DFHJCR *DFHJCTTE *DFHLFM *DFHTCADS *DFHTCTFX	
DFHJCIOE	DFHJCINA	DFHJCP	The journal control I/O error program is called by a journal task that discovers an I/O error in journal output. If the JCT entry is flagged retry, then DFHJCIOE switches to a new output volume and returns to let the journal task retry the output. For tape journals, DFHJCIOE treats a unit exception condition as end-of-reel and sets the EOV flag and returns. If an I/O error occurs immediately, or if the JCT was not flagged retry then DFHJCIOE writes message DFH4513 to the operator and abends the journal task. If the journal is specified as crucial, the operator is requested to terminate CICS. For standard-labeled tapes, DFHJCIOE calls DFHVCP to record any failures. For systems with data sharing, DFHJCIOE calls DFHDLI to tell it of the defective tape.	DFHDLI DFHJCP DFHPCP DFHSCP DFHTDP DFHVCP	*DFHCSADS *DFHJCADS DFHJCICA *DFHJCTDS *DFHJCTTE *DFHTCADS DFHTDOA	
DFHJCJFP	DFHJCJNA	Op sys	The journal control formatting program is a separate, stand-alone program that journal data set. DFHJCJFP writes a preformatted record on each track, which allows CICS to locate and reposition to the most recently written record on a disk journal whenever CICS is restarted.	None	DFHJCICA *DFHJCR	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCKOJ	DFHJCKNA	DFHSIP	The journal control kickoff program is called from DFHSIP and runs under terminal control's TCA. DFHJCKOJ performs the following functions: • Establishes the start time, used for journal tape volume identification. • Attaches a CICS task whose purpose is to wait on the termination of journaling's open/close operating system subtask. • For standard-labeled tape journals, resolves the addresses of exit routines, and stores them in the JCT. • For standard-labeled tape journals, calls DFHVCP to select the first volume of each journal to write on. • For standard-labeled journal tapes, calls DFHVCP to enter a first volume ID if the series is empty. • For each journal in the JCT, DFHJCKOJ opens the journal for output (if not specified as deferred), attaches the CICS journal task for it, and waits until that journal task is in position. • Issues a message to the console saying how many journals have been successfully opened.	DFHABEND DFHBSP DFHJC DFHJCP DFHKCP DFHPCP DFHSCP DFHVCP DFHWTO	*DFHCSADS *DFHJCADS *DFHJCOCL *DFHJCTDS *DFHJCTTE *DFHSIT *DFHTCA DFHVOLDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCO	DFHJCONA	DFHJCP	The journal control open program processes all journal data set OPEN requests. DFHJCO: • Identifies the standard-labeled volumes and data sets that DFHJCOCP operates on. • Checks the status of journals. • Initializes a journal for input or output. • Issues pause messages and waits if replies have not been received. • Validates the volume request. • Performs or sets up file positioning, including journals in the SMF format. • Performs or sets up any device/data set switch implied by the combination of request type and volume request. • If opening for output and journaling to to disk, calls DFHXJCO just before passing the request to the subtask. • Enqueues to a single thread on an OS/VS POST used in communicating with the open/close subtask, DFHJCOCP. • Issues a CICS WAIT for the requesting task on the ECB posted by the subtask on completion. • Dequeues. • Updates the condition, chaining, and time stamps on standard-labeled volumes as required. • Calls DFHWKP to update the restart record in the restart data set with the IDs of new volumes, if they are standard-labeled tapes. • Sets start-of-volume and pseudo-buffer flags as appropriate. • For systems with data sharing, calls DFHDLI to inform DBRC of the change in the system log. • Returns to the calling program.	DFHCCP DFHDLI DFHJCOCP DFHKCP DFHPCP DFHSCP DFHVCP DFHWTO DFHXJCO	*DFHCSADS *DFHDWEDS *DFHJCA DFHJCICA DFHJCOCL *DFHJCR DFHJCSMF *DFHJCTDS *DFHJCTTE *DFHKPPDS *DFHLFM *DFHMGM *DFHPCTDS DFHSAADS *DFHTCA *DFHTCTFX DFHVOLDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCOCP	DFHJCONA	DFHJCC DFHJCO	The journal control open/close program runs as an OS/VS subtask and allows the main CICS task to remain dispatchable while journals are being opened or closed. DFHJCOCP issues appropriate MOUNT or UNLOAD messages to the processing unit (CPU) console if necessary. The program runs as a subtask and therefore does not return to a caller but detaches itself when requested. The interface between the requestor and DFHJCOCP is a WAIT/POST interface, which has the effect of eventually returning control to the requestor, but not through program exit. Standard-labeled journal tapes are identified by the calling parameters, and the operating system performs open or close without operator intervention. This module holds exit routines for operating system use.	None	*DFHAFCD *DFHCSADS DFHJCADS DFHJCOCL *DFHJCR DFHJCSMF *DFHJCTDS *DFHJCTTE *DFHSIT *DFHTCTFX DFHTULDS	Op sys
DFHJCP	DFHJCPNA	DFHJC DFHSIP	The journal control program: • Analyzes all requests, and links to other modules for OPEN, CLOSE, GETB, GETF, NOTE, and POINT requests. It performs WRITE, WAIT, and PUT requests itself. • On its first write to a new data set, writes the CICS monitoring facility dictionaries, using the pseudo-buffer mechanism. • Processes all DFHJC requests and may cause the user task that called to wait if necessary. • Performs output event-completion processing running under one of the journal tasks (one per JCT entry). The journal task for a journal is resumed whenever a user task starts input or output for the journal, or waits on some event on it. The resumed journal task for the journal starts input or output if necessary, waits on its completion, shifts up the journal buffer if appropriate, and then restarts any waiting user tasks. DFHJCP checks any replies received from pause messages for the current journal. It then checks if it has any more work to do; if not, it suspends the task under which it is running until resumed by a subsequent user task.	DFHCCP DFHICP DFHJCBSP DFHJCC DFHJCEOV DFHJCI DFHJCIOE DFHJCO DFHJCSDJ DFHKCP DFHPCP DFHSCP DFHTSP DFHVCP	*DFHCSADS DFHDBLDS DFHDBRDS DFHDLP *DFHDMEDS *DFHJCA DFHJCICA DFHJCOCL *DFHJCR DFHJCSMF *DFHJCTDS *DFHJCTTE *DFHLFM *DFHPCTDS *DFHSIT *DFHTCA *DFHTCTFX *DFHTCTTE *DFHUEXIT	
DFHJCRP	DFHJCRNA	DFHSII1	The journal control recovery program is called during initialization to restore journal control states from information in the catalog.	DFHCCP DFHJCP	*DFHCSAPS *DFHDLPDS *DFHJCTDS *DFHJCTTE *DFHTCAPS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHJCSDJ	DFHJCSNA	DFHJCBSP DFHJCEOV DFHSTP	The journal control shutdown journaling program is called by: • DFHJCEOV when a crucial journal incurs an error on end-of-volume switching. • DFHSTP when CICS is closing down. • DFHJCBSP when the operating system open/close subtask terminates abnormally. DFHJCSDJ shuts down all journaling activity as follows: • Enqueues for each journal in turn. • Closes each open journal. • Terminates each journal task. • Terminates the open/close subtask. • Issues a message to the console operator, dequeues, and returns to the calling program.	DFHJCP DFHKCP DFHPCP DFHWTO	*DFHCSADS *DFHJCADS DFHJCOCL *DFHJCTDS *DFHJCTTE *DFHTCA	
DFHJUP	DFHJUPNA		The journal print utility program examines, selects, and displays data in QSAM data sets, such as the CICS and IMS/VS logs. Data selection is controlled by input parameters, and an optional user exit	None	None	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHKCP	DFHKCPNA	DFHKC (macro)	The task control program has the following main functions: • To service DFHKC macros under control of a requesting task's TCA. • To select the next task to be given control (task dispatcher function). • To issue the system wait macro when there is nothing for CICS to do. • To add or delete entries in the PCT. DFHKCP also contains the CICS time-of-day routine, the task tuning logic (for example, MXT, AMXT, and CMXT), and part of the runaway task. Listed below are the main subroutines of DFHKCP: KCDC - Task dispatcher KCDCTR - Dispatch control KCDEQN - Dequeue from a resource KCDQALN - Dequeue all resources KCENQN - Enqueue upon a resource KCPATSK - Purgeable task scan KCPC - Change priority of a task KCTDDTN - Return to a task KCTDDDTN - Dispatch a task KCTDWT - Release control to operating system KCTO - Attach a task KCTRN - Resume a task KCTSN - Suspend a task KCWAIT - DFHKC TYPE=WAIT support	DFHCMP DFHDIP DFHICP DFHKCP DFHPCP DFHSCP DFHSPP DFHTCP DFHTRP DFHUEM DFHZCB DFHZCX	*DFHAIDDS *DFHCRBDS *DFHCSADS DFHDCADS *DFHDCTDS *DFHEIS *DFHICEDS *DFHIRSDS *DFHISCRQ DFHKCTWA *DFHLFM DFHLLADS *DFHPAMDS *DFHPCTDS *DFHQCADS DFHQEADS DFHSAADS *DFHTCADS DFHTCTCE	Op sys Task to be dispatched Caller
DFHKCRP	DFHKCRP		DFHKCRP is the task control restart program.	DFHCCP DFHLFA DFHPCP DFHRCP	DFHCCRQ *DFHCSADS *DFHLFM *DFHPCTDS *DFHSIPDS *DFHSITDS DFHRCRQ *DFHTCA	
DFHKCSP	DFHKCSPA DFHKCSPI DFHKCSPL DFHKCSPD DFHKCSPF DFHLCSPP	None	The task SRB control program is part of the High Performance Option code available on CICS on MVS. It runs in SRB mode and resides in protected storage.	None	*DFHCSADS DFHDCADS DFHHTADS *DFHTCA	
DFHLFA	DFHLFNA		The LIFO prolog routine is included in the CSA and performs the LIFO stack initialization required by any LIFO invocation. It formats the LIFO stack entry, and optionally makes a trace entry for the invoking module.	DFHLFO	None	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHLFO	DFHLFONA	DFHLFA	The LIFO stack overflow routine handles the situation when a request for LIFO storage exceeds the size of the current LIFO stack; a new stack is obtained and all relevant pointers are updated. DFHLFO also handles the freeing of the LIFO segments.	DFHSCP DFHTRP	*DFHCSADS *DFHLFM *DFHPCTDS *DFHTCA	
DFHLUP	DFHLUPNA		DFHLUP is the LU6.2 services manager. It initializes and shuts down a network, and resynchronizes flows.	DFHICP DFHLFO DFHPCP DFHSCP	DFHCSADS DFHLUCDS DFHTCADS DFHZEPD	
DFHMCP	DFHMCPNA	User appl	The mapping control program processes DFHBMS macro requests and completes the processing of a logical message when a task terminates without issuing a DFHBMS TYPE=PAGEOUT. DFHMCP's main function is to analyze DFHBMS requests and to pass control to the appropriate modules. Other functions include the loading of maps and partition sets, and scheduling of output messages transmitted by temporary storage. Listed below are the main subsections of DFHMCP: MCPCPO - Complete logical message build message control record for temporary storage. MCPDWEXT - DWE processing, invoked by DFHKCP to complete BMS processing at application termination. MCPINPT - Handles all input requests. MCPIN - TYPE=IN (EXEC CICS RECEIVE MAP) MCPMAPLO - Subroutine loads map set and locates the map. MCPPGBLD - TYPE=PAGEBLD or TEXTBLD or EXEC SEND TEXT MCPPGOUT - TYPE=PAGEOUT (EXEC CICS SEND PAGE) MCPPURGE - TYPE=PURGE (EXEC CICS PURGE MESSAGE) MCPROUTE - TYPE=ROUTE (EXEC CICS ROUTE)	DFHICP DFHIIP DFHM32 DFHPBP DFHPCP DFHPLR DFHSCP DFHTCP DFHTDP DFHTPQ DFHTPR DFHTPS DFHTRP	*DFHCSADS *DFHDWEDS *DFHICEDS *DFHLFM DFHMAPDS *DFHMCRDS DFHOSPWA *DFHPPTDS DFHSAADS *DFHSIT *DFHTCA *DFHTCTZE *DFHTRACE *DFHTTPDS	
DFHMCX	DFHMCXNA	DFHMCP	DFHMCX is the BMS fast path module for standard and full BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions: • It is a noncumulative direct terminal send map or receive map issued by a command-level program. • It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state. • The CSPQ transaction has been started. • The message disposition has not changed.	DFHLFO DFHPCP DFHSCP DFHTCP DFHZCP	DFHMSCA DFHCSADS DFHMAPDS DFHTCADS DFHTCTZE DFHTIOA	
DFHMDLSG			DFHMDLSG is a program which is an example of how to invoke the DFHSG macro to add DL/I support.	DFHSG		

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHMGP	DFHMGPNA	DFHACP DFHSPP DFHZEMW DFHZNAC	The message interface module provides an interface for sending messages to transient data destinations CSMT, CSCS, and CSTL, to the console, or to a terminal. The input to DFHMGP consists of the binary number of the message to be produced together with any information to be inserted. DFHMGP builds a complete message from a prototype held in DFHMGT, and sends it to the appropriate destination. The prototype statements are invocations of the DFHMGM TYPE=TEXT macro, and are contained in copy books held in DFHMGT. Normally it would only be necessary to change DFHMGT (not DFHMGP) to change the format and/or text of a message.	DFHLFO DFHPCP DFHSCP DFHTCP DFHTDP DFHTRP DFHXSP DFHZCP	*DFHCSADS *DFHDCTDS *DFHLFM *DFHMGM *DFHPCTDS *DFHSNT *DFHTACB *DFHTCA *DFHTCPCM DFHTCTFX *DFHTCTZE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHMGT	DFHMGTNA	DFHMGP	The message prototype control table consists of a series of copy books, DFHMGTnn, each of which contains up to 100 messages. They are arranged such that the copy book DFHMGTnn contains messages DFHnnxx, where xx = 00-99. For example, DFH2314 is in copy book DFHMGT23. Within each copy book are invocations of DFHMGM (in ascending message number order) for messages which are currently in use with CICS. However, the inclusion of the message within DFHMGT does not necessarily mean that the actual message produced by CICS will use the MGP/MGT process. Provided the message is produced by the MGP/MGT process then the message can be altered by changing the text in the appropriate copy book, and then assembling DFHMGT. The main operands of the macro DFHMGM TYPE=TEXT are: • MSGNO = actual message number • MG1 = prototype message components • MG2-9 = as MG1, being a means of allowing concatenation The prototype message components are one of the following: • Characters enclosed in quotes are to be reproduced unchanged in the message • IIII = insert into the message (the character strings provided by the module calling MGP) • TRANS or PPPP = insert the transaction name • TERM or TTTT = insert the terminal name • ABCD = insert the original (first) abend code • Time = insert the time • Date = insert the date Other inserts do exist but are more specialized. DFHMGT insists that all messages have a time stamp, so that if the TIME option is not specified in the prototype, a time stamp is added at the end of the message.	Not applic	Not applic	Not applic

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHMIR	DFHMIRNA	DFHISP	The mirror program is invoked when a request to access a resource is received from a remote system (through ISC or MRO). DFHMIR may be thought of as returning the answer to the requesting actions of DFHISP. DFHMIR controls the receipt of requests and transmission of replies. It also determines if protected resources have been modified in which case it continues receiving requests and transmitting replies until a sync point request is received from the remote system. DFHMIR invokes DFHXFP or DFHXFX to analyze the requests that have been received and to prepare the replies for transmission. DFHMIR also passes the request to DFHEIP, DFHDLI or DL/I for execution. A mirror task on an IRC link suspends itself on completion of a request and it is then be available for use by any other MRO function shipped request. DFHKCP terminates the mirror task if it is not reused within two seconds.	DFHDLI DFHEIP DFHISP DFHKCP DFHPCP DFHSPP DFHTCP DFHTRP DFHXFP DFHXFX DL/I	*DFHCSADS *DFHDWEDS *DFHLFM *DFHTCA *DFHTCTTE *DFHXFSTG	
DFHML1	DFHML1NA	DFHMCP DFHPBP	The SCSPRT logical unit type 1 output mapping routine is called by DFHPBP to build a page of data stream from a chain of map and application data structure copies. The data contains only features that the TTP says are supported by the target terminal. This routine is called when NLEOM is specified for 3270 printers or LU3 printers. Listed below are the main subsections of DFHML1: ML1SPACE - Calculate space for chaining and mapping. ML1FMCA - Format the chains that describe the maps. ML1PF - Process map fields.	DFHPCP DFHSCP DFHTCP	*DFHCSAS DFHMAPDS *DFHMCADS DFHOSPWA DFHPGADS *DFHTCA *DFHTCTTE DFHTIOA *DFHTTPDS	
DFHMSP	DFHMSPNA	CMSG trans	The message switching program is invoked by the CMSG transaction. DFHMSP's purpose is to route a message entered at the terminal to one or more operator-defined terminals or to other operators. DFHMSP can be used in conversational mode to process operands entered from separate input operations. In this case the operands already processed are preserved on temporary storage. Listed below are the main sections, and subroutines of DFHMSP: MSBMSRT - Check for complete operands MSCNVRS - Issue conversational response MSCONTIN - Process conversational response MSMSG4 - MSG operand MSNTRY - Process operands MSROUTE - Route operand	DFHMCP DFHPCP DFHSCP DFHTCP DFHTSP	*DFHCSADS *DFHPCTDS *DFHPPTDS *DFHTCA *DFHTCTTE DFHTIOA DFHTSIOA DFHURLDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHMTPA	DFHMTPNA	CSMT CSOT CSST CWTR	A master terminal routine that is loaded for CSMT, CSOT, CSST and CWTR transactions and performs the first analysis of operands. DFHMTPA handles inquiries about or changes to: • AKP frequency • Runaway task time interval, and handles CSMT SNAP requests DFHMTPA transfers control to: • One of the other DFHMTP modules to deal with other operands, or • DFHSTP for shutdown requests. This program is also used when NLEOM is specified for 3270 printers or LU3 printers.	DFHFDP DFHICP DFHMTPF DFHPCP DFHSCP	*DFHCSADS *DFHLFM DFHMTWM *DFHPCTDS *DFHSIT *DFHTCA DFHTCTLE *DFHTCTTE DFHTIOA	DFHMTPB DFHMTPC DFHMTPD DFHMTPE DFHMTPF DFHMTPG DFHSTP
DFHMTPB	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for enquiries about or changes to the following master terminal options: • Cushion size • Maximum tasks • Negative poll delay • Maximum number of active tasks allowed • Maximum tasks in a class of tasks, and set normal trace on or off.	DFHFCP DFHPCP DFHSCP DFHTCP DFHTRP	*DFHCSADS *DFHFCTDS *DFHLFM *DFHMTWM DFHOCLDS *DFHTCA DFHTCTLE *DFHTCTTE DFHTIOA	DFHMTPA DFHMTPD User
DFHMTPC	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for the master terminal options that inquire about or change the status of terminals.	DFHISP DFHKCP DFHPCP DFHSCP DFHTCP DFHZARL DFHZLOC DFHZLUS DFHZSTY	*DFHCSADS DFHDCADS *DFHISCRQ *DFHLFM DFHMTWM *DFHPCTDS *DFHPPTDS *DFHTCA DFHTCTLE *DFHTCTZE DFHTIOA	User
DFHMTPD	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for the following master terminal options: • OPEN and CLOSE data sets (data base, transient data, extrapartition, and dump). • SWITCH dump data sets.	DFHFCP DFHICP DFHPCP DFHSCP DFHTDP	*DFHCSADS *DFHDCTDS *DFHFCTDS DFHFCTSR *DFHLFM DFHMTWM DFHOCLDS DFHOCODS *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTTE DFHTIOA DFHZEPD	DFHMTPB User

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHMTPE	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for the following master terminal options: • TRIGGER LEVEL • NEW COPY • STALL • AUXILIARY TRACE	DFHICP DFHMTPA DFHPCP DFHSCP DFHTDP DFHTRP	*DFHCSADS *DFHDCTDS *DFHLFM DFHMTWM *DFHPPTDS *DFHTCA DFHTCTLE *DFHTCTZE DFHTIOA DFHZEPD	User
DFHMTPF	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for the following master terminal options: • LINE • CNTRL • TERMINAT • TASKL	DFHKCP DFHPCP DFHSCP DFHTCP	*DFHCSADS DFHDCADS *DFHDCTDS *DFHLFM DFHMTWM *DFHPCTDS *DFHPPTDS DFHSAADS *DFHTCA *DFHTCTLE DFHTIOA DFHTCTTE	DFHMTPC User
DFHMTPG	DFHMTPNA	DFHMTPA	A master terminal routine that receives control from DFHMTPA for the options that enable or disable a transaction, program, data base data set, or extrapartition data set.	DFHKCP DFHPCP DFHSCP	*DFHCSADS *DFHLFM DFHMTWM *DFHPCTDS *DFHPPTDS *DFHTCTLE DFHTCTTE DFHTIOA	DFHMTPB DFHMTPD DFHMTPE User
DFHMXP	DFHMXPNA	Auto task init-iation	The local queuing shipper provides the means of transferring to a remote system, a START request which has been temporarily deferred by use of the local queuing option.	Not applic	DFHFMHDS	
DFHM32	DFHM32NA	DFHMCP DFHPBP	For a BMS output request, the 3270 mapping program generates the appropriate data stream for a 3270 device, and returns control to DFHPBP which invokes module DFHTPP to send the data to the appropriate destination, which is either to the direct terminal, to temporary storage or back to the caller. For a BMS input request, the data stream from a 3270 device is examined, and mapped into a user application TIOA format. Listed below are the main subsections of DFHM32: BMFMHTST - Create beginning of 3270 data stream (FMH cursor positioning) BMMID - Input mapping BMMMS - Merge maps (output mapping) M32PF - Process field.	DFHPCP DFHSCP DFHTCP	*DFHCSADS DFHMAPDS *DFHMCAD DFHOSPWA DFHPGADS *DFHTCA *DFHTCTTE DFHTIOA *DFHTTPDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHOCP	DFHOCPNA	Dump control user trans	The dynamic open/close program opens or closes transaction dump data sets, transient data extrapartition data sets, and file control data sets. DFHOCP can be invoked either from an application program, or from the master terminal program.	DFHFCP DFHKCP DFHPCP DFHSCP DFHTDP	*DFHAFCD *DFHCSADS *DFHDCO *DFHDCTDS *DFHFCTDS *DFHLFM DFHOCLDS DFHOCODS *DFHPPTDS *DFHTCA	
DFHPBP	DFHPBPNA	DFHMCP	The page and text build program positions maps or text, including header or trailer maps or text, within a page of output. For non-3270 devices the module creates a page buffer containing the users data which is then passed to DFHDSB to produce a device dependent data stream. When mapping, this includes merging the data supplied by the application with the constant data included in the map. For 3270 devices, copies of the maps and application supplied data for a page are chained together, to be processed by module DFHM32, to produce a 3270 data stream. The page and text build program creates dummy maps, and chains them in the same way for 3270 text building. For LU1 printers with extended attributes, copies of the maps and application supplied data for a page are chained together, to be processed by module DFHML1 to produce an SCS data stream. The page and text build program creates dummy maps, and chains them in the same way for text building. After the maps have been processed by DFHDSB, DFHM32 or DFHML1, DFHPBP calls DFHTPP to write them out.	DFHDSB DFHML1 DFHM32 DFHPCP DFHSCP DFHTPP	*DFHCSADS DFHMAPDS DFHOSPWA DFHSLDC *DFHTCA *DFHTCTTE DFHTIOA *DFHTTPDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO

Listed below are the main subroutines of DFHPBP.

PBDOUTPT - Mapping/textbuild complete, decide whether to call data stream generator and which one (DFHDSB or DFHM32). Return to caller (DFHMCP).

PBD00005 - Main control logic, request analysis.

PBD01000 - Map placement logic (3270 and non-3270 mapping).

PBD01130 - Non-3270 mapping.

PBD10000 - Pageout routine.

PBD11000 - Modify field positions within map (used by 3270 and non-3270 mapping).

PBD20000 - Text processing (3270 and non-3270).

PBD30000 - 3270 mapping.

PBFMHBLD - Build FMH if FMHPARM specified (non-3270 text and map processing).

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHPCP	DFHPCNNA	DFHPC macro	The program control program controls the execution of all programs defined in the PPT, and has three main functions: • To service DFHPC macros. • To provide an interface between PL/I or COBOL application programs using the macro-level interface and CICS and for all programs that use the command-level interface to CICS. The interface functions are provided by DFHEIP. • To add or delete entries in the PPT. Listed below are the main subroutines of DFHPCP: PCABEND - Build transaction abend control block and abnormally terminate a task PCBLDL - Get directory information about a program if the BLDL slot has not been initialized PCBLIN - COBOL interface PCDLPN - Delete loaded program PCCLOAD - Loader communication PCCPRRN - Program release PCDELETE - Delete request processing PCLINK - Pass program control anticipating return PCPF - Program fetch PCPL1IN - PL/I interface PCPPTSN - PPT search PCRETRY - Restart task after abend PCSETXIT - Cancel/activate/reactivate an Abend exit.	DFHACP DFHDBP DFHDCP DFHICP DFHKCP DFHPCP DFHPPT DFHSCP DFHSPP DFHTCP DFHTRP	*DFHCSADS DFHDCADS *DFHEIS *DFHJCA *DFHLFM DFHLLADS *DFHPAMDS *DFHPCTDS *DFHPPTDS *DFHSCCOS *DFHTACB *DFHTCA *DFHTCTTE *DFHTCTZE *DFHTRACE *DFHUEXIT *DFHZEPD	COBOL pgm PL/I pgm
DFHPCRP	DFHPCRPN		DFHPCRP is the program control restart program.	DFHCCP DFHLFA DFHPCP DFHRCP DFHTMP	DFHCCRQ *DFHCSADS *DFHLFM *DFHPPTDS DFHRCRQ *DFHSIPDS *DFHSIT *DFHTCA DFHTMRQ	
DFHPDXRF	DFHPDXRF	DFHPDX1	Part of the PRDMP exit. It formats control blocks relating to XRF support, including CAVM trace.	PRDMP services	DFHWCGPS DFHWCSPS DFHWDGPS DFHWFGPS DFHWMGPS DFHWSAPS DFHWSXPS DFHWTGPS	
DFHPDX1	DFHPDXNA	MVS/XA PRDMP program	Runs as an exit from the MVS/XA PRDMP program. Formats an SVC or stand-alone dump using PRDMP services to extract data and print output, including interpreted CICS trace.	DFHPDXRF PRDMP services	DFHDSCTS	
DFHPDX2	DFHPDX2	DFHPDX1	Part of PRDMP exit. Serves as a vehicle for DFHTUREN, the CICS trace formatter.		DFHTUREN	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHPEP	DFHPEPNA	DFHACP	The program error program is IBM-supplied and establishes a base register, establishes addressability to the system portion of the TCA, and returns control to DFHACP. DFHPEP can be modified by the user to perform further recovery operations.	DFHPCP	*DFHCSADS *DFHPCTDS *DFHTCA	DFHACP
DFHPHP	DFHPHPNA	DFHMCP DFHTOM	The partition handling program has one entry point, and starts with a branch table which passes control to the required routine according to the request. Listed below are the routines in this module: PHPPSI - Loads a partition set. PHPPSC - Destroys any existing partitions and creates new partitions. PHPPIN - Extracts the AID, cursor position, and partition ID. PHPPXE - Activates the appropriate partition if data is received from an unexpected partition.	DFHLFM DFHPCP DFHSCP DFHTCP	DFHCSADS DFHOSPWA DFHPGADS DFHPSDDS DFHTCA DFHTCTZE DFHTIOA DFHTPE	
DFHPL1OI			The PL/I interface module contains the following routines: DFHPL1N - Initial entry point for PL/I PL/I programs under CICS DFHPL1I - CICS macro service interface DFHPL1C - Set the CSA address IBMBOCLA/B/C - Bootstraps for open/close functions	DFHPCP DFHSAP	*DFHCSADS *DFHTCA	
DFHPRK	DFHPRKNA	DFHZATT	The 3270 print key program (CSPK) is invoked when, under VTAM, the 3270 program access key designated as the print key is depressed and no task is attached to the terminal. If the 3270 hardware copy feature is present, DFHPRK attaches task CSCY to the printer designated in the TCTTE, and DFHCPY is executed. If the copy feature is not present, DFHPRK executes a DFHTC TYPE=PRINT macro.	DFHICP DFHPCP DFHSCP DFHTCP	*DFHCSADS *DFHTCA *DFHTCTZE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHPRPR	DFHPRPNA	Not applic	The high-level language preprocessor program is an offline utility program that prepares COBOL and PL/I application programs (written with CICS macros) for macro expansion and compilation. Listed below are the main subroutines of DFHPRPR: CICS - CICS macro EOD - Terminate preprocessor INITIMG - Convert initial image NOTCICS - High-level language OUTPUT - Output routine PRSTRIPN - Strip character PRSTRIPX - Terminate strip READ - Read and scan input WRITEC - Write and get next record.	None	None	Not applic
DFHPSP	DFHPSPNA	DFHEPS	DFHPSP is the system spooling interface control module.	DFHPSPSS	*DFHCSADS DFHPSPPS *DFHTCA	
DFHPSPDW	DFHPSPDW	DFHSPP	DFHPSPDW is the system spooling interface deferred work element (DWE) processor.		*DFHCSADS *DFHDWEDS DFHPSGPS *DFHTCA	
DFHPSPSS	DFHPSPSS	DFHPSP	The system spooling interface subtask module attaches a subtask to check that a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.	DFHPSPST DFHPSSVC DFHSKP	DFHAFCD *DFHCSADS DFHPDBPS DFHPSGPS DFHPSVPS IEFZB4D0 IEFZB4D2	
DFHPSPST	DFHPSPST	DFHPSPSS	DFHPSPST is the system spooling interface control module.	DFHPSSVC	*DFHCSADS *DFHDWEDS DFHPDBPS DFHPSGPS *DFHTCA	
DFHPSPVC	DFHPSSNA	DFHPSPSS DFHPSPST	DFHPSSVC is the system spooling interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.		DFHAFCD DFHAFCS *DFHSECDS IEFJESCT IEFJSSOB IEFJSSIB IEFJSSSO IEFZB4D0 IEFZB4D2 IKJTCB IHAPSA IHARB IHASCB IHASDWA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHPUP	DFHPUPNA	DFHAMP DFHCSDUP	The parameter utility program transforms the definition data of the CSD. In the CSD, the data is held in a compacted form and each field is self-identifying. Elsewhere in the processing, these fields are handled in parameterized form, using an argument address list. It also serves to transform the resource definition to the original high-level command.			
DFHP3270	DFHP32NA	CSPP trans DFHTCP DFHZCP	The 3270 print program prints 3270 data received from a screen on a 3270 printer. The data is compressed where possible and then transmitted to the printer.	DFHICP DFHPCP DFHSCP DFHTCP	*DFHCSADS *DFHTCA *DFHTCTTE DFHTIOA DFHTSIOA	
DFHQRY	DFHQRY	DFHALP DFHTCTI DFHZATT	The query transaction (DFHQRY) sends a READ PARTITION QUERY structured field to a 3270, analyzes the response, and completes information in the corresponding TCTTE. DFHQRY can be attached by DFHALP, DFHTCTI, or DFHZATT.	DFHZCQ	DFHBMSCA *DFHCSADS DFHEIBLK *DFHTCA *DFHTCTTE	
DFHRCEX	DFHRCEX		DFHRCEX enables the global user exits for emergency restart processing.	DFHLFA	*DFHCSADS DFHRSADS *DFHSIT *DFHTCA	
DFHRCP	DFHRCPNA		The recovery control program: Opens a recovery file Closes a recovery file Establishes a thread Releases a thread Starts a browse Ends a browse Writes a record to the recovery file Reads a record from the recovery file Gets the next record from the recovery file Deletes a record from the recovery file Deletes all records of a specified type Writes a record to the log Initializes recovery processing Performs restart processing.	DFHLFA	*DFHCSADS *DFHJCADS *DFHLFM *DFHTCA IFGACB IFGRPL	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHRCRP	DFHRCRP		DFHRCRP is the recovery control restart program.	DFHLFA DFHPCP DFHRCP DFHRUP	DFHCCRQ *DFHCSADS *DFHJCTDS *DFHJCTTE *DFHMRCPS DFHRCRQ *DFHSIPDS *DFHSIT *DFHTCA *DFHTDSDS *DFHTSMDS	
DFHRKB	DFHRKBNA	DFHCPY	The release 3270 keyboard program is initiated by DFHCPY to release a 3270 keyboard. It does this by issuing a DFHTC TYPE=WRITE macro which sends a 3270 write control character.	DFHPCP DFHSCP DFHTCP DFHZCP	*DFHCSADS *DFHTCA *DFHTCTZE DFHTIOA	CICS
DFHRLR	DFHRLRNA	DFHMCP	The route list resolution program builds a TTP (terminal type parameter) control block for each type of terminal for which a message is to be built. A TTP is acquired for each terminal type in the user route list and the direct terminal if there is one. Listed below are the main subsections of DFHRLR: RLRALL - Routing with ROUTE=ALL RLRLIST - Routing with route list specified in application RLROPLL - Routing with OPCALSS in RLRRTEBY - Nonrouting, non-LDC device (that is direct terminal) RLR3601 - Nonrouting LDC device.	DFHPCP DFHSCP DFHTCP	*DFHCSADS DFHOSPWA DFHSAADS DFHSLDC *DFHTCA *DFHTCTZE *DFHTTPDS DFHURLDS	
DFHRMSY	DFHRMSY	DFHSPP	The purpose of task-related user exit resynchronization is to resolve any in-doubt LUWs. Task-related user exit resynchronization is called by sync point management during execution of the RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.	DFHEIP	DFHURD	
DFHRTE	DFHRTENA	DFHPCP	The transaction routing program establishes a transaction routing session with a remote region specified by the user. Subsequent input is analyzed by DFHRTE, and the transcode extracted, and a request issued to DFHZTSP to route the transaction to the required system.	DFHMGP DFHPCP DFHTSP DFHZARQ DFHZLOC DFHZTSP	*DFHCSADS *DFHISCRQ *DFHLFM *DFHMGM *DFHPCTDS *DFHTCA *DFHTCPCM DFHTCTFX *DFHTCTZE DFHTIOA DFHZEPD	
DFHRTY	DFHRTYNA	DFHDBP	The IBM-supplied retry program establishes a base register and addressability to the system part of the TCA, and issues a message to destination CSMT. DFHRTY can be modified by the user to perform logical tests which decide whether or not a transaction is to be restarted.	DFHPCP DFHSCP DFHTDP	*DFHCSADS *DFHPCTDS *DFHTCA DFHTDOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHRUP	DFHRUPNA	DFHRCRP	The recovery utility program is invoked during emergency restart. It extracts current information from the system log and saves it on the recovery file for accessing by DFHRCP during backout of in-flight tasks. It also passes all volume data to DFHVCP. Listed below are the main subroutines of DFHRUP: RUDLIPRR - DL/I update RUINITBR - Initialization RULABELR - Label processing RUPRDBDR - Data base update RUPRKPR - Keypoint update RUPRLOGR - Process data base user key record RUPRSOTR - Start of task processing RUPRUSER - User update RURDLOGR - Read system log RUSYNCHR - Sync update RUTERMR - Termination RUUPDCTR - DCT update RUUTSTR - Update TST RUWRMSGR - Message processing RUWRRSDR - Write restart data set RUWRSINR - Statistics processing.	DFHJCP DFHKCP DFHPCP DFHRCP DFHSCP DFHTCP DFHTDP DFHTDRP DFHTSRP DFHVCP	*DFHCSADS *DFHDBO DFHDCT *DFHDCTDS *DFHDLP *DFHFBO *DFHJCA *DFHJCRDS DFHKPTE *DFHMBO *DFHSIT *DFHTBO *DFHTCA *DFHTCTZE DFHTDCI DFHTURD DFHVOLDS	
DFHRWP70	DFHRW70	DFHTCP	The 7770 read/write program issues a read initial, write continue, write conversational, or write disconnect as required to a 7770 device.	None	DFHTCTLE	DFHTCP
DFHSCP	DFHSCPNA	API	The main function of the storage control program is to handle all storage requests and to return the address of acquired storage in the TCA; DFHSCP also controls the short-on-storage indicator in the CSA. If a request for storage is invalid DFHSCP issues a DFHPC TYPE=ABEND macro to terminate the task; if an unconditional request for storage cannot be satisfied, DFHSCP issues a DFHKC TYPE=SUSPEND macro to suspend the task until the request can be satisfied. Listed below are the main subroutines of DFHSCP: SCCTLFRE - Control subpool (FREEMAIN) SCCTLGBF - Control subpool (GETMAIN) SCPGETM - Controller (GETMAIN) SCPGMFRE - Program subpool (FREEMAIN) SCPGMGET - Program subpool (GETMAIN) SCSHRFRE - Shared subpool (FREEMAIN) SCSHRGBF - Shared subpool (GETMAIN) SCTELFRE - Teleprocessing subpool (FREEMAIN) SCTELGET - Teleprocessing subpool (GETMAIN) SCTSKFRE - Task subpool (FREEMAIN) SCTSKGET - Task subpool (GETMAIN).	DFHKCP DFHPCP DFHTRP	*DFHCSADS DFHDCADS *DFHPAMDS *DFHPCTDS *DFHPPTDS DFHSAADS *DFHSCCOS *DFHTCA *DFHTCTTE *DFHTCTZE *DFHTRACE *DFHUEXIT	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSCR	DFHSCRNA	DFHSCP DFHSCR	The storage control recovery program is invoked either when DFHSCP detects a storage violation, or if a program check occurs in DFHSCP. DFHSCR verifies storage chains and FAQE chains and attempts to correct errors, for example by rewriting an SAA or by repairing a forward chain from pointer information in a backward chain. DFHSCR can invoke a storage violation (formatted) dump.	DFHFDP DFHTRP	*DFHCSADS *DFHPAMDS *DFHPCTDS *DFHPPTDS *DFHSCCOS *DFHTCA *DFHTCTTE	
DFHSCTE	DFHSCTNA	Not applic	The subsystem control table extension resides in the link pack area (LPA). DFHSCTE is used by the interregion communication program (DFHIRP) as an anchor for its control block structure. DFHSCTE contains a doubleword which is updated by means of compare-and-swap-double (CDS) to ensure integrity. The first word contains the address of the logon address control block (LACB) and the second contains a count of the number of users who have logged on to the interregion program. DFHSCTE does not contain any executable code.	None	None	Not applic
DFHSFP	DFHSFPNA	DFHSNP	The sign-off program gets control through DFHSNP either in response to a CSSF sign-off request, or if a user tries to sign on while still signed on for a previous session. DFHSFP resets internal CICS flags and sends session data to the statistics file. If the external security manager (RACF) is active, the user control block (ACEE) is freed.	DFHTDP DFHXSP	*DFHCSADS *DFHLFM *DFHSEC *DFHTCA DFHTCTLE *DFHTCTTE	DFHSNP
DFHSIA1	DFHSIANA	DFHSIP	A system initialization program that deals with override parameters.	None	*DFHSICOM	
DFHSIB1	DFHSIBNA	DFHSIP	A system initialization program that loads the CICS nucleus, loads and initializes the CSA, and builds the trace table.	DFHSIP	*DFHDLP *DFHJCTDS *DFHJCTTE DFHKPPDS *DFHPCTDS *DFHSICOM *DFHTCA DFHTCTFX *DFHTRACE DFHZEPD	
DFHSIC1	DFHSICNA	DFHSIP	A system initialization program that initializes the user exit table, if required. DFHSIC1 opens the restart data set, and determines what type of start to execute. It also attaches DFHTEOF, if required.	DFHSIP	*DFHPCTDS *DFHSICOM *DFHUEXIT	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSID1	DFHSIDNA	DFHSIP	A system initialization program which performs the following functions: • Initializes the transient data facility. • Opens transient data data sets. • Performs a warm start for the transient data facility. • Initializes intrapartition storage. • Initializes multiple buffers for transient data.	DFHSIP	*DFHDCTDS DFHKPDDS DFHMBCDS DFHMRCDS DFHSICOM DFHTDCI DFHTDSDS *DFHVSWA	
DFHSIE1	DFHSIENA	DFHSIP	A system initialization program that: • Calculates and builds the VSAM LSR pool. DFHSIE1 performs this only if DL/I is specified, and there is no multiple LSR pool support. In all other cases, DFHFCL builds the pools.	DFHSIP	*DFHFCTDS DFHFCTSR *DFHSICOM	
DFHSIF1	DFHSIFNA	DFHSIP	A system initialization program that initializes terminal control. DFHSIF1: • Opens BTAM terminal control data sets. • Opens the VTAM ACB. • Builds hash-table entries for non-VTAM terminals. • Constructs a DFHZCP module list in the TCT prefix. • Initializes the attach tables.	DFHLFA DFHSIP	*DFHAFCD *DFHLFM *DFHPCTDS *DFHSICOM *DFHTCA DFHTCTFX DFHTCTLE DFHTCTSK *DFHZAIT DFHZEPD	
DFHSIG1	DFHSIGNA	DFHSIP	A system initialization program that: • Opens the dump data set. • Initializes temporary storage.	DFHDCP DFHSIP DFHXSP	*DFHDCO *DFHDCTDS *DFHFCTDS *DFHJCTTE *DFHPCTDS *DFHSICOM *DFHTCTZE DFHTSCTL DFHTSTDS	
DFHSIH1	DFHSIHNA	DFHSIP	A system initialization program that: • Attaches the journal control subtask. • Builds temporary storage and transient data bit maps. • Builds the page allocation map. • Calculates OSCOR. • Calls DFHDLQ to initialize DL/I. • Initializes multiple buffers for temporary storage.	DFHSIP	DFHALTDS DFHJCOCL *DFHJCTDS DFHKCTWA *DFHPAMDS *DFHSICOM *DFHTCA DFHTSBDS *DFHTSMDS *DFHVSWA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSII1	DFHSIINA	DFHSIP	A system initialization program that establishes program check and abnormal termination exits in DFHSRP. DFHSII1 performs warm start functions for temporary storage, ICEs and AIDs, and CSA. DFHSII1: • Initializes auxiliary trace. • Attaches the CICS restart tasks to run in parallel: Recovery control Terminal control Program control Task control Transient data. Security interface DL/I File control Temporary storage • Waits for the restart tasks to complete. • Processes the GRPLIST parameter. • Allocates storage for resident application programs. • Warm starts the CSA, AIDs, ICEs, and temporary storage. • Takes an activity keypoint. • For emergency restart, asks the operator if initialization is to continue. • Initializes system console support.	DFHALP DFHAMP DFHFDP DFHICP DFHPCP DFHSCP DFHSIP DFHSRP DFHTMP DFHTRP DFHTSP	*DFHAIDDS *DFHDCTDS *DFHICEDS *DFHFCTDS DFHFMIDS DFHKCTDS DFHKPDDS *DFHPCTDS *DFHPPTDS *DFHSICOM *DFHTCA *DFHTCTZE DFHTMRQ DFHTSBDS DFHTSMDS DFHWREDS DFHZEPD	
DFHSIJ1	DFHSIJNA	DFHSIP	This program is the last to be executed in the process of system initialization. DFHSIJ1 opens and preformats the restart data set, executes the programs listed in DFHPLT and then passes control to DFHTCP. On emergency restart, DFHSIJI links to DFHRCP. For standard-labeled tapes, DFHSIJ1 calls DFHVCP to create the series definition table, and to initialize the system log series with the volume it needs for emergency restart, if emergency restart is being performed. DFHSIJ1 calls DFHDLX with the parameter '*PIN' to complete the initialization for DL/I. DFHSIJ1 calls the VS COBOL II partition initialization routine, passing parameters.	DFHDLX DFHICP DFHPCP DFHSCP DFHSIP DFHSPP DFHTCP DFHVCP	DFHDCADS DFHJCOCL DFHKPPDS *DFHLFM DFHPLTDS *DFHSICOM *DFHTCA *DFHTCTZE *DFHUEXIT	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSIP	DFHSIPNA	Op sys	The main function of the system initialization program is to initialize CICS. DFHSIP also provides DFHWTO support, and common subroutines used by DFHSIA1 through DFHSIJ1 In sequence, DFHSIP performs the following functions: • Receives control from the operating system. • Scans the override parameters for SIT and CSA suffixes. • Loads the appropriate SIT and CSA. • Passes control to the routines DFHSIA1, DFHSIB1, etc. Listed below are the main subroutines in DFHSIP: OVERLSUP - Overlay supervisor SIGTMAIN - Startup SIPCONS - Console WRITE.	DFHFDP	*DFHKPBUF DFHKPPDS *DFHSICOM *DFHSIPD	Not applic
DFHSKP	DFHSKMNA DFHSKC DFHSKE		DFHSKP consists of three modules: DFHSKM - the subtask manager DFHSKC - the subtask control program DFHSKE - the subtask execution program. DFHSKM calls and, if necessary, attaches DFHSKC to process the created work queue element (WQE). DFHSKM also causes termination of the subtask when requested and handles DWE processing and task cancel requests. DFHSKC starts an operating system subtask, DFHSKE, and waits for its completion. DFHSKE processes WQEs, looking at in-progress and waiting queues on a first-in, fist-out basis. DFHSKE intercepts program checks and operating system abends.			
DFHSNP	DFHSNPNA	CSSF CSSM 8888 9999 trans	The sign-on program is called in response to a CSSN sign-on request. DFHSNP checks the user's identification and password against the sign-on table; if authorized, the user is signed on. If the security program is active, DFHSNP interprets the sign-on parameters, prompts the operator for more parameters if needed, and passes the values to the security manager for verification.	DFHPCP DFHSCP DFHTCP DFHXSP	*DFHCSADS *DFHLFM *DFHREGS *DFHSEC *DFHSNT *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTZE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSPP	DFHSPPNA	User appl DFHKCP DFHZNAC	The sync point program is invoked during a user-specified sync point (by a DFHSP macro) or at task termination. For a rollback request only, DFHSPP calls DFHDBP to restore recoverable resources. It scans the DWE chain invoking the appropriate DWE processors, and performs the necessary sync point logging. It dequeues all resources enqueued by the transaction. For an ISC environment, DFHSPP calls DFHPSZ to transmit a sync-point request to the linked systems to synchronize remote resource changes. DFHSPP processes any DWEs connected with the resource manager, and processes the RESYNC command. Listed below are the main subroutines of DFHSPP: SPP00005 - Write DWE log data SPP02020 - Build a DWE chain that can be logged. SPP03000 - End.	DFHEIP DFHICP DFHJCP DFHKCP DFHPCP DFHSCP DFHSPZ DFHSXP DFHTCP DFHTSP	*DFHCSADS DFHDBLDS DFHDBRDS DFHDCADS *DFHDLIAL *DFHDWEDS *DFHISCRQ *DFHJCA *DFHLFM *DFHMGM *DFHTCA *DFHTCTZE DFHURD DFHZEPD	
DFHSRP	DFHSRPNA	Op sys	The system recovery program deals with program check interrupts and system abends. In the event of a program check, DFHSRP passes control to DFHPCP as if for a DFHPC TYPE=ABEND, ABCODE=ASRA macro. In the event of a system abend, DFHSRP searches the SRT for the abend code that has arisen and, if it is found, invokes the user-supplied program specified. Afterwards DFHSRP can either abend CICS or attempt to keep it running with only the faulty task abended (ASRB). Listed below are the main subroutines of DFHSRP: SRPSTRTR - STAE retry SRPSTRTR - ESTAE retry	DFHDLI DFHEIT DFHFDP DFHIRP DFHPCP DFHSCR DFHTDP DFHTRP DFHTSP DFHXFP	*DFHAFCD *DFHCSADS *DFHDLP DFHKPPDS *DFHPCTDS *DFHPPTDS DFHSAADS *DFHSIT DFHSRTDS *DFHSTR *DFHTCA	OS/VS
DFHSTKC	DFHSTKNA	None	The supervisory statistics program writes task and storage statistics to a transient data destination, usually CSSL. DFHSTKC may be invoked in response to a CSTT statistics request from a terminal or by DFHSTP at system shutdown. For further statistics it links either to DFHSTSP for automatic statistics (CSTT AUT) or DFHSTPF for requested or shutdown statistics (CSTT AOR, or CEMT and CSMT SHUT).	DFHPCP DFHSCP DFHTCP DFHTDP	*DFHCSADS *DFHLFM *DFHPAMDS *DFHPCTDS DFHSAADS *DFHSIT *DFHTCA DFHTCTLE *DFHTCTZE DFHTIOA	DFHSTPD DFHSTSP
DFHSTLK	DFHSTLNA	DFHSTTR	The link statistics program is called by DFHSTTR and records statistics related to ISC links, and IRC links (for MRO and DL/I shared data base).	DFHPCP DFHTDP	*DFHCSADS *DFHLFM DFHSAADS *DFHTCA DFHTCTFX *DFHTCTZE	DFHSTTD

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSTP	DFHSTPNA	DFHMTPA	The main function of the system termination program is to shut down CICS. In sequence, DFHSTP performs the following functions (according to options specified): 1. Loads DFHXLT and DFHPLT. 2. Quiesces the terminals. 3. Executes the programs defined in the first part of DFHPLT. 4. Closes the automatic statistics data set (links to DFHSTSP), cancels ICEs, waits for DFHTCP to service all output requests, and breaks storage chains. 5. Executes the programs defined in the second part of DFHPLT. 6. Takes statistics (links to DFHSTKC), flushes the temporary storage buffer, cancels the program check exit, cancels abnormal termination exit (if there is one). 7. Calls DFHVCP to delete those volume descriptions that are of no further use. 8. Closes down DL/I facilities by calling DFHDLI. 9. Shuts down journal control (links to DFHJCSDJ). 10. For systems with data sharing, calls DFHDLI to shut down DBRC and IRLM. 11. Closes auxiliary trace, takes a warm keypoint if required. 12. Takes a formatted dump (if required). 13. Passes control to the operating system. Listed below are the main subroutines of DFHSTP: First stage quiesce 1 STASKQ - Determine type of shutdown 2 STXLTLD - Load transaction list table (XLT). 3 STPLTLD - Load program list table (PLT) 4 STPLTLNK - Execute programs in PLT 5 STBREAK - Cancel exits and break storage chains 6 STPIRCBP - Quiesce terminals Second stage quiesce 7 STPLT2 - Execute PLT program 8 STSHUDLI - Close DL/I and start final processing 9 STKPEND - Take dump if requested 10 STSTATBY - Close transient data and temporary storage STCHAPO - Subroutine to change task status	DFHCMON DFHDLI DFHFCP DFHFDP DFHICP DFHIRP DFHISP DFHJCSDJ DFHKCP DFHPCP DFHSCP DFHSPP DFHSTKC DFHTCP DFHTDP DFHTRP DFHTSP DFHTSP DFHWKP Programs in the PLT and XLT	DFHAFCD *DFHAIDDS *DFHCRBDS *DFHCSADS DFHDCADS *DFHDCO DFHDLP *DFHIRSDS *DFHISCRQ *DFHJCTDS *DFHJCTTE *DFHSIT *DFHTCA DFHTCTCE DFHTCTFX DFHTCTLE *DFHTCTZE DFHTIOA DFHVOLDS DFHZEPD	Op sys

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
			During CICS initialization with START=EMER after an abnormal termination, if the operator replies "CANCEL" to a request to go or not go, DFHSTP is called. The switch in STFLAGS is set to STCINIT to select the code to be executed. During CICS initialization with START=LOGTERM, DFHSTP is called to write an end-of-file in the system log for later offline processing of the log.			
DFHSTPD	DFHSTPNA	DFHSTKC	The transaction, program, and dump statistics program is called by DFHSTKC. DFHSTPD collects statistics from the PPT, PCT, and CSA (for dumps), and writes them, for subsequent printout, to the transient data destination whose name is passed by DFHSTKC in the CWA.	DFHPCP DFHTDP	*DFHCSADS *DFHLFM *DFHPCTDS *DFHPPTDS DFHSAADS *DFHTCA DFHTCTLE *DFHTCTTE DFHTIOA	DFHSTTR
DFHSTSP	DFHSTSNA	DFHSTKC	The automatic statistics program, can be called either from a terminal CSTT AUT transaction (initially, using DFHSTKC) or automatically at preset intervals from CICS. CSTT is used to start or to stop automatic statistics recording, or to switch from one recording file to another. For automatic statistics recording, an ICE is set up which recalls DFHSTSP periodically to take a snapshot of statistical data. DFHSTSP puts statistical data to a transient data destination, CSSM or CSSN, whichever is active. It is usual to make these extrapartition destinations with file names DFHSTM and DFHSTN.	DFHFCP DFHICP DFHKCP DFHOCP DFHPCP DFHSCP DFHTDP	*DFHCSADS *DFHDCTDS *DFHFCTDS DFHFCTSR *DFHJCTDS *DFHJCTTE *DFHLFM *DFHPAMDS *DFHPCTDS *DFHPPTDS *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTZE DFHTIOA *DFHTSMDS	CICS
DFHSTTD	DFHSTTDN	DFHSTLK	The data management statistics program is called by DFHSTLK, and records statistics relating to transient data, temporary storage and journaling.	DFHPCP DFHTDP	*DFHCSADS *DFHDCTDS *DFHJCTDS *DFHJCTTE *DFHLFM DFHSAADS *DFHTCA *DFHTSMDS	CICS
DFHSTTR	DFHSTTNA	DFHSTPD	The file and terminal statistics program is called by DFHSTPD and handles terminal and file statistics, including VSAM shared resources statistics. DFHSTTR writes statistics to a transient data destination for later access or printout by the user. The transient data destination is chosen by the user, but defaults to CSSL.	DFHOCP DFHPCP DFHSCP DFHTDP	*DFHCSADS *DFHFCTDS DFHFCTSR DFHSAADS *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTZE	DFHSTLK

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHSTUP	DFHSTUNA	Op sys	The statistics utility program prints the automatic statistics data set. There are two options, a report by intervals or a summary report. DFHSTUP passes a 1-byte parameter to a separate routine to perform printing. If the parameter is invalid, the routine abends with the PSW pointing to the 4-byte string with the characters "BADI".	None	None	Op sys
DFHSVCSG			DFHSVCSG is a program which is an example of how to invoke the DFHSG macro to change the SVC numbers used by CICS.	DFHSG		
DFHTACP	DFHTACPN	DFHTCP	The terminal abnormal condition program is invoked by DFHTCP and performs the following functions: • Analyzes error codes in the TACLE. • Sends appropriate messages to the CSMT transient data destination (for terminal errors) or to the CSTL transient data destination (for logical errors). • Invokes the user supplied (or sample) terminal error program (DFHTEP). • Takes the appropriate actions resulting from the defaults which may have been modified by the terminal error program.	DFHICP DFHKCP DFHPCP DFHSCP DFHTDP DFHTRP	*DFHCSADS DFHDCADS *DFHPCTDS *DFHTACLE *DFHTCA DFHTCTLE DFHTDOA DFHTIOA	DFHTCP
DFHTAJP	DFHTAJNA		The time adjustment program calls DFHICP to reset the CSA's time fields according to the host-supplied time-of-day. DFHTAJP then scans the ICE chain and adjusts the expiry time of interval-controlled ICEs. Time-controlled ICEs are not adjusted but the ICE chain is reordered so that it is left in order by expiry time. Times held in the TCT and CSATCNDT are decreased, and negative times are made zero. Lastly, DFHTAJP writes a message.	DFHICP DFHPCP	*DFHCSADS *DFHICEDS *DFHTCADS DFHTCTFX DFHTCTLE *DFHTCTZE	
DFHTBSB	DFHTBSB		DFHTBSB adds a node to the control-block structure. It is called during the dynamic installation of TCT resources, and calls routines in the control block builder.			
DFHTBSBP	DFHTBSBP	DFHTBS00 DFHTBSBP	DFHTBSBP is the recursive part of DFHTBSB.			
DFHTBSD	DFHTBSD		DFHTBSD deletes a node in a CICS terminal network.			
DFHTBSDP	DFHTBSDP	DFHTBSD DFHTBSDP	DFHTBSDP is the recursive part of DFHTBSD.			

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTBSL	DFHTBSL		DFHTBSL creates the recovery record for a node during the dynamic installation of a TCT table entry using the CEDA INSTALL command, for example, and calls routines in the control-block builder.			
DFHTBSLP	DFHTBSLP	DFHTBSL DFHTBSLP	DFHTBSLP is the recursive part of DFHTBSL.			
DFHTBSQ	DFHTBSQ		DFHTBSQ is called to retrieve the parameters that were supplied to a TCT table entry at build time.			
DFHTBSQP	DFHTBSQP	DFHTBSQ	DFHTBSQP is called by DFHTBSQ to retrieve parameters that were supplied to a TCT table entry at build time.			
DFHTBSR	DFHTBSR		DFHTBSR takes a table-builder recovery record and re-creates the corresponding table entry. It is called during WARM or EMERGENCY restart.			
DFHTBSRP	DFHTBSRP	DFHTBSR	DFHTBSRP is called by DFHTBSR.			
DFHTBSSP	DFHTBSSP		DFHTBSSP performs a commit or rollback action for a previous table-builder change according to the outcome of a logical unit of work. Each action is dequeued from a deferred work element (DWE).			
DFHTBSOO	DFHTBS		DFHTBSOO is the main routine for DFHTBS and holds the addresses of the modules used to build control blocks for the dynamic installation of TCT resources.			
DFHTCBP	DFHTCBNA		The terminal control backout program restores TCTTEs and other ISC state data during emergency restart.	DFHPCP DFHRCEX DFHRCP DFHSCP DFHTSP DFHZLOC	*DFHCSADS DFHFMIDS *DFHIMSDS *DFHJCRDS *DFHLFM *DFHMBO DFHRCRQ *DFHSIT *DFHTCA *DFHTCTZE *DFHUEXIT DFHURDDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTCP	DFHTCPNA		Terminal control program. The terminal control task is attached during system initialization and remains until termination. DFHTCP manages all non-VTAM terminals, which involves: • Ensuring that I/O operations are started when possible on the lines • Analyzing completion information • Attaching transactions when data is received from a terminal and no task is attached to that terminal • Servicing terminal control requests from user transactions. Listed below are the modules and subsections of DFHTCP: DFHTCCBS - Terminal control common BSC routines DFHTCCLC - Terminal control line control scan routine DFHTCCOM - Terminal control common logic DFHTCCP - Terminal control compatibility DFHTCCSS - Terminal control start-stop common logic DFHTCDEF - Terminal control symbol definition DFHTCNBS - Terminal control nonswitched BSC DFHTCN29 - Terminal control nonswitched 2980 device dependent DFHTCN36 - Terminal control nonswitched 3600 device dependent DFHTCN70 - Terminal control nonswitched 2770 device dependent DFHTCN74 - Terminal control nonswitched 3740 device dependent DFHTCN80 - Terminal control nonswitched 2780 device dependent DFHTCORS - Terminal control storage handling DFHTCSAM - Terminal control sequential terminal device dependent DFHTCSBS - Terminal control common switched BSC routines DFHTCSNC - Terminal control start-stop non-dial common routines DFHTCSSC - Terminal control start-stop switched common routines DFHTCS35 - Terminal control switched 3735 device dependent DFHTCS7N - Terminal control nonswitched System/7 device dependent DFHTCS70 - Terminal control switched 2770 device dependent DFHTCS74 - Terminal control switched 3740 device dependent DFHTCS80 - Terminal control switched 2780 device-dependent DFHTCTI - Terminal control task initiation	DFHICP DFHKCP DFHPCP DFHSCP DFHTACP	*DFHCSADS *DFHDCTDS *DFHPCTDS *DFHSIT *DFHTCA DFHTCTFX DFHTCTLE *DFHTCTTE DFHTIOA DFHXLTDS DFHZEPD	CICS

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
			DFHTCTLX - Terminal control world trade teletype device-dependent DFHTCTRN - Terminal control translate tables DFHTCTWX - Terminal control TWX device-dependent DFHTC37D - Terminal control switched 3270 device-dependent DFHTC40N - Terminal control nonswitched 2740 device-dependent DFHTC40S - Terminal control switched 2740 device-dependent DFHTC41N - Terminal control nonswitched 2741 device-dependent module DFHTC41S - Terminal control switched 2741 device-dependent DFHTC50N - Terminal control nonswitched 1050 device-dependent DFHTC50S - Terminal control switched 1050 device-dependent DFHTC70C - Terminal control common 3270 subroutines DFHTC70L - Terminal control local 3270 device-dependent DFHTC70R - Terminal control remote 3270 device-dependent DFHTC77S - Terminal control 7770 device-dependent.			
DFHTCRP	DFHTCRPA		DFHTCRP initializes and recovers terminal control definitions and protected messages. It is run as a task during CICS initialization.	DFHCCP DFHLFA DFHRCP DFHWTO	*DFHCSADS *DFHLFM *DFHSIPDS *DFHSIT *DFHTCA *DFHTCTFX *DFHTCTSK *DFHTCTTE DFHZCPLN	
DFHTCRPC	DFHTCRPC	DFHZCQO	XRF tracking interface for TCT contents. One of a set of routines called by DFHZXQO from the same CALL statement, the entry-point address having been passed to DFHZXQO. This routine calls ZC RESTORE to add or delete a TCT entry based on information from another CICS system using the log, the catalog, or the XRF tracking queues.	DFHZCQPS		
DFHTCRPG	DFHTCRPG	DFHTCRP	Processes the ZCP key range of the log to recover in-flight committed ZC INSTALLs.			
DFHTCRPL			Installs TCT resources defined by the TCT macros.			
DFHTCRPS	DFHTCRPS	DFHZCQO	XRF tracking interface for ZCP sessions. One of a set of routines called by DFHZXQO from the same CALL statement, the entry-point address having been passed to DFHZXQO. This routine calls DFHZXST (through DFHZXS) to make changes to the session state. DFHWCCS	DFHZXST		

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTDP	DFHTDPNA	User appl DFHELR DFHETD DFHMGP DFHRUP DFHTDRP DFHTPQ DFHTPR DFHTSRP DFHZNAC	The transient data program services DFHTD requests. Most of its function is concerned with intrapartition queues where it is responsible for: • Keeping track of the current input and output positions. • Space management on the auxiliary storage device. • Automatic task initiation when trigger levels are reached. • Managing multiple buffers and multiple buffer strings. Listed below are the main sections of DFHTDP: REQANAL - Request analysis TDEXTR - Extrapartition processing EXTRAGET - GET processing EXTRAPUT - PUT processing EXTRAFEV - FEOV processing TDINTR - Intrapartition processing INTRAGET - GET processing INTRAPUT - PUT processing INTRAPUR - PURGE processing TDPRDWE - DWE processing	DFHJCP DFHKCP DFHPCP DFHSCP DFHTDP DFHVSP	*DFHAIDDS *DFHCSADS DFHDCT *DFHDCTDS *DFHDWEDS *DFHTCA DFHTDCI DFHTDIA DFHTDOA *DFHUEXIT	
DFHTDRP	DFHTDRNA	DFHTDX	The transient data recovery program is invoked by execution of a DFHTD TYPE=INITIALIZE macro. DFHTDRP builds the DCT index from the DCT load module. For a warm start, it restores the DCT and CI state map from the keypoint data that was saved earlier. For an emergency restart, it restores the CI state map from the DCT entries that were initialized by DFHRUP using information in the system log.	DFHKCP DFHPCP DFHSCP DFHTDP	*DFHCSADS DFHDCT *DFHDCTDS *DFHTCA DFHTDCI DFHTDIA	
DFHTEOF	DFHTEONA	Op sys DFHSIC1	The tape-end-of-file program can be executed as part of system initialization if the system log created during a previous CICS run is on tape. It writes an EOF mark following the last valid record written prior to an abend of the previous run, by reading the tape forward and comparing the label records in each record. For standard-labeled journal tapes, the ID of the tape is specified by the calling parameters, or by the JCL.	None	*DFHJCR DFHJCSMF *DFHSIPD DFHTULDS	
DFHTEP	DFHTEPNA	DFHTACP	The terminal error program is invoked by DFHTACP by a DFHPC TYPE=LINK macro. The sample DFHTEP (invoked only if there is no customer-supplied version) puts a terminal out of service if the number of terminal errors detected by DFHTACP exceeds default values contained in DFHTEP tables.	DFHPCP	*DFHCSADS *DFHTACLE *DFHTCADS DFHTCTLE *DFHTCTTE DFHTDOA	DFHTACP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTMP	DFHTM macro		The table management program performs locates, adds, deletes, locks and unlocks to entries in certain CICS tables. DFHTMP uses a hash table for these operations. Listed below are the main subroutines of DFHTMP: CMTORBKO - Commit or back out table changes COMMIT - Commit ENCHAIN - Enchain LOCATETE - Locate TMLOCK - Set a read lock CHKTTC - Validate table type code LOCFDIRE - Locate a free directory element GETMSTG - Get shared storage FRETMSTG - Free shared storage GETUSTG - Get user storage FRETUSTG - Free user storage ENQUEUE - Enqueue on table modifications DEQUEUE - Dequeue on table modifications ENQDE - Enqueue directory element DEQDE - Dequeue directory element CRTCLE - Create change list element CRTDWE - Create a deferred work element SRCHRL - Note read lock SRCHRLT - Search for read lock in chain.			
DFHTON	DFHTONNA	DFHDBP DFHSPP	The terminal object resolution module is called by DFHDBP or DFHSPP during DWE processing for DFHTOR. It calls DFHTOR with end-LUW-cancel or end-LUW-commit code to perform cancel or commit of changes to TERMINAL, TYPETERM, CONNECTIONS, or SESSIONS definitions.	DFHTOR	*DFHCSADS *DFHDWEDS *DFHTCA	
DFHTOR	DFHTORNA	DFHAMP DFHTON	DFHTOR is the terminal object resolution program. DFHAMP calls DFHTOR for a TERMINAL, TYPETERM, CONNECTION, or SESSION object in a CICS system definition (CSD) file that is being installed, or when DFHAMP encounters an end-of-group. DFHTOR processes the objects and passes them to the terminal control builder program (DFHZCQ). The DFHTON entry is used for DWE processing.	DFHCCP DFHRCP DFHSCP	*DFHCSADS *DFHDWEDS *DFHTCA	
DFHTORP	DFHTORNA	DFHSII1	DFHTORP is the terminal object recovery program. It is called during CICS initialization to purge TYPETERM and model terminal definitions from the catalog on a cold start, and to recover these definitions on an emergency restart.	DFHCCP DFHRCP	*DFHSIT	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTPP	DFHTPPNA	DFHDSB DFHM32	The terminal page processor program handles DFHBMS TYPE=OUT, STORE and RETURN requests. If OUT, DFHTPP sends the complete page using DFHTC macro requests; if STORE, the page is sent to temporary storage; and if RETURN, no output operation takes place but the page is returned to the application program. Listed below are the main subroutines of DFHTPP: TPNODDS - TYPE=STORE (PAGING) requests TPOUT - TYPE=OUT (TERMINAL) requests (the macro DFHTOM is used by both DFHTPP and DFHTPR to handle output to terminals.) TPRETPG - TYPE=RETURN (SET) requests.	DFHPCP DFHSCP DFHTCP DFHTDP DFHTSP	*DFHCSADS *DFHDWEDS DFHFMHDS DFHOSPWA DFHPGADS *DFHTCA DFHTCTLE *DFHTCTZE DFHTIOA DFHTSIOA *DFHTTPDS	DFHPBP
DFHTPQ	DFHTPQNA	DFHICP DFHMCP DFHTCP	The undelivered messages cleanup program is initiated periodically in order to cancel the delivery of BMS messages which have been placed on temporary storage, but have remained undelivered for an interval exceeding the purge delay time specified in the generation of BMS.	DFHALP DFHICP DFHKCP DFHMCP DFHPCP DFHSCP DFHTCP DFHTDP DFHTSP	*DFHAIDDS *DFHAL *DFHCSADS *DFHLFM *DFHMCRDS DFHOSPWA *DFHSIT *DFHTCA *DFHTCTZE DFHTDOA DFHTIOA DFHURLDS DFHZEPD	CICS using DFHPCP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTPR	DFHTPRNA	DFHMCP DFHPCP DFHTCP	The terminal page retrieval program (CSPG) is invoked: • By automatic transaction initiation as a result of a SCHEDULE issued by DFHTPS. • By a DFHPC LINK from DFHMCP, when CTRL= RETAIN or RELEASE on DFHBMS = PAGEOUT (RETAIN or RELEASE on SEND PAGE in command-level). • When CSPG or an operator paging command is entered at a terminal. If the message is autopaged, DFHTPR retrieves the pages of the message in order, transmits them to the terminal, and then purges the message. Otherwise DFHTPR runs pseudo-conversationally. All further input is passed to DFHTPR, until the message is purged explicitly or implicitly. If the input is a valid paging command (page retrieval, page copy, page purge or page chaining) it is processed. It is rejected if explicit purge is required or passed back to normal task initiation, if automatic purge is allowed. Listed below are the main subsections and subroutines of DFHTPR: DFHMSPUT - Send error message to terminal TPENCCHN - Encode and execute page chain TPENCCOP - Encode and execute page copy TPENCPUR - Execute page purge TPENCRET - Encode page retrieval TPERETA - Reset to autopaging TPERETQ - Page query TPEXIT - Exit from program TPEXPUR - Execute page purge TPEXRET - Execute page retrieval TPTSGET - Get MCR or page from temporary storage.	DFHALP DFHBMS DFHICP DFHKCP DFHLFM DFHMCP DFHPCP DFHSCP DFHTCP DFHTDP DFHTRP DFHTSP	*DFHAIDDS *DFHAL *DFHCSADS DFHFMHDS *DFHMCBDS *DFHMCRDS DFHOSPWA DFHPGADS *DFHSIT DFHSLDC *DFHTCA DFHTCTLE *DFHTCTZE DFHTDOA DFHTIOA DFHTSIOA *DFHTTPDS DFHURLDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTPS	DFHTPSNA	DFHICP DFHMCP DFHPCP	The terminal page scheduling program (CSPS) is invoked for each terminal type to which a BMS logical message built with TYPE=STORE is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status. DFHTPS contains the following four major subsections to deal with four separate functions: • DFHTPSNA - DFHTPS is invoked by automatic initiation on expiry of ICE, and as a result of an IC PUT request issued by DFHMCP (there is no associated terminal). This invocation SCHEDULES CSPG for terminals on this system and SCHEDULES CSPS on the link to each remote system which owns terminals contained in the route list for the message (that is the function of TPS02000). • TPS01000 - DFHTPS is linked to from DFHMCP for direct paging requests to a terminal on a remote system. The task has a surrogate TCTTE as its primary facility, and owns a relay link connected to the terminal owning system. This section ships the pages of the message to the terminal owning system, where it will be re-created by the relay program (DFHCRP) which issues BMS, STORE, TEXT, NOEDIT and PAGEOUT requests. • TPS02000 - DFHTPS is scheduled to run against the link to a remote system (by TPS01000). This routine ships the logical message to the remote system and deletes the terminals on the remote system from the terminal list in the original message control record (TPS03000 receives the information at the remote system). • TPS03000 - DFHTPS is invoked by an ATTACH request from a remote system (that is originated by TPS01000 or TPS02000). This receives the shipped logical message and issues BMS, ROUTE, TEXTBLD, NOEDIT, and PAGEOUT requests to re-create the logical message on the terminal owning system. DFHTPS contains the following subroutine: • TPSSHIPM - ship a complete logical message.	DFHICP DFHISP DFHKCP DFHMCP DFHMGP DFHPCP DFHSCP DFHSPP DFHTDP DFHTSP DFHXTP DFHZARQ DFHZLOC DFHZTSP	*DFHAIDDS *DFHCSADS *DFHISCRQ DFHLFM *DFHMCRDS *DFHMGM DFHOSPWA *DFHTCA *DFHTCTZE DFHTIOA DFHTSIOA DFHURLDS *DFHXTSTG DFHZEPD	CICS through DFHPCP
DFHTRAP	DFHTRANA	DFHTRP	The FE global trap/trace exit is provided for diagnostic use only under the guidance of service personnel.	None	*DFHTRACE DFHTRADS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTRP	DFHTRPNA	DFHCMP	The trace program maintains an in-storage trace table whose entries identify different types of system activity. Time-stamped trace entries can also be maintained on an auxiliary data set. Listed below are the main subroutines of DFHTRP: TACLOSE - Close auxiliary trace TAOPEN - Open auxiliary trace TRAUXPUT - Write auxiliary trace record TREXIT - Exit used after building trace entry DFHTRPFQ - FAQE chain checker	DFHTRAP	*DFHAFCD *DFHAIDDS *DFHAL *DFHCSADS DFHDCADS *DFHDCTDS DFHFWADS DFHISCRQ *DFHJCADS *DFHLFM *DFHPAMDS *DFHPPTDS *DFHSC *DFHSEC *DFHTCA *DFHTCTCE DFHTCTFX *DFHTCTZE *DFHTRACE DFHTRADS *DFHVSWA *DFHXFSTG	
DFHTRZCP	DFHTRZCP	CEDA trans DFHTCRP DFHTOR	DFHTRZCP builds a terminal builder parameter set.		DFHZCQPS	
DFHTRZIP	DFHTRZIP	CEDA trans DFHTCRP DFHTOR	DFHTRZIP builds a chain of builder parameter sets for sessions.		DFHZCQPS	
DFHTRZPP	DFHTRZPP	CEDA trans DFHTCRP DFHTOR	DFHTRZPP builds a pool builder parameter set.		DFHZCQPS	
DFHTRZXP	DFHTRZXP	CEDA trans DFHTCRP DFHTOR	DFHTRZXP builds a connection builder parameter set.		DFHZCQPS	
DFHTRZYP	DFHTRZYP	CEDA trans DFHTCRP DFHTOR	DFHTRZYP builds a typeterm builder parameter set.		DFHZCQPS	
DFHTRZZP	DFHTRZZP	CEDA trans DFHTCRP DFHTOR	DFHTRZZP merges a TYPETERM builder parameter set into a terminal builder parameter set.		DFHZCQPS	
DFHTSBP	DFHTSBNA		The temporary storage backout program restores the recoverable temporary storage REWRITEs during emergency restart.	DFHLFA	*DFHCSADS DFHFMIDS *DFHJCRDS *DFHLFM DFHRCRQ *DFHTCA *DFHTSCI	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHTSP	DFHTSPNA	User applic DFHDBP DFHDIP DFHETS DFHICP DFHJCP DFHKCP DFHMCP DFHMSP DFHSAMP DFHSII1 DFHSRP DFHSTP DFHTPQ DFHTPR DFHTSRP DFHZCP DFHZRSP	The temporary storage program services DFHTS requests. It maintains the tables, directories and maps necessary to keep track of every temporary storage record and of available space on the VSAM auxiliary storage or in main storage. Listed below are the main subroutines of DFHTSP: TSP020 - Store data TSP020 - Retrieve data TSP020 - Release data DFHTSPAM - Manage auxiliary storage (including multiple buffers and strings)	DFHJCP DFHKCP DFHPCP DFHSCP DFHTSP DFHVSP	*DFHCSADS DFHDCADS *DFHDWEDS *DFHICEDS *DFHJCADS *DFHLFM DFHSIT *DFHTCA DFHTSCI DFHTSBPS *DFHTSMDS DFHTSMPS DFHTSTDS *DFHUEXIT DFHVSCAS *DFHVSWA DFHVSWAS	
DFHTSRP	DFHTSRNA	DFHRUP	The temporary storage recovery program recovers the temporary storage tables, directories, and maps when it is invoked by DFHRUP during emergency restart. To do this, it uses the information recovered from the system log by DFHRUP and the contents of the temporary storage VSAM data set itself.	DFHICP DFHKCP DFHPCP DFHSCP DFHTDP DFHTSP	*DFHCSADS *DFHICEDS *DFHTCA DFHTSCI *DFHTSMDS	
DFHTUP			The trace utility program formats and prints trace records stored on the auxiliary trace data set. This utility program is run as a separate job and extracts selected trace entries as specified on parameter cards supplied as part of the input to the program		*DFHTCA	
DFHTUREN		DFHDUP DFHFDP DFHTUP	The trace utility routine decodes and formats one trace table entry ready for printing. DFHTUREN is used by DFHFDP, DFHDUP and DFHTUP.		None	CICS
DFHTXRP	DFHTXRP	DFHTCRP	DFHTXRP installs macro-defined TCT entries.		*DFHCSADS *DFHTCA *DFHTCTTE	
DFHTXRPR	DFHTXRPR	DFHTCRP	DFHTXRPR installs nonmigrated TCT entries.		*DFHCSADS *DFHTCA *DFHTCTTE DFHZCQPS	
DFHUEH	DFHUEHNA	CICS management modules containing exit points	The user exit handler is a link between a CICS management module exit point and user code. DFHUEH invokes each started exit program for that exit, passing a parameter list and on return updates the return code for the management module.	User exit program	*DFHCSADS *DFHLFM *DFHTCA *DFHTRACE *DFHUEXIT	
DFHUEM	DFHUEMNA	DFHEIP	The EXEC interface processor is for user exit commands. For further information on this module refer to DFHEIP.	DFHPCP	*DFHEIPPL *DFHUEXIT	DFHEIP

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHUSBP	DFHUSBNA		The user backout program sends user log records to a user exit during emergency restart. The records are extracted by DFHRUP from the restart data set. They are in the system log for in-flight logical units of work.	DFHLFA	*DFHCSADS DFHFMIDS *DFHJCRDS *DFHLFM *DFHSIT *DFHTBODS *DFHTCA *DFHTSCI *DFHUEXIT	
DFHVAP	DFHVAP	DFHSIJ1	If using CICS VSAM subtasking, DFHVAP is a program that attaches the OS/VS subtask DFHVSP and waits for it to terminate or abend If DFHVSP abends, so does DFHVAP.	DFHICP DFHKCP DFHPCP DFHSCP	DFHCSADS DFHTCADS DFHVSCA	
DFHVCP	DFHVCPNA	DFHJCC DFHJCEOV DFHJCKOJ DFHJCO DFHSII1	The volume management program provides services for CICS management modules. DFHVCP keeps track of the data volumes in use by, or reserved for, other facilities using information passed to it by journal control, the master terminal operators, and the terminal operators. DFHVCP tells its caller which volumes to use for each OPEN request or when a volume switch occurs. It also gives requested information to the master terminal program.	DFHSCP DFHTRP	DFHCSADS DFHLFM DFHOFL DFHSDT DFHTCADS DFHTUL DFHVOLDS	
DFHVSP	DFHVSPNA	OS/VS	The CICS VSAM/BSAM subtask program DFHVSP: • Issues a SPIE to handle program checks. • Issues an OS/VS WAIT for posting by DFHFCP or completion of a request. • Processes any SQEs on the request queue. • Posts CICS if required.	DFHSCP	DFHCSADS DFHSQES DFHTACB DFHTCADS DFHVSCA	
DFHWCCS	DFHWCCS	Many CAVM modules	Common services for CAVM: MVS FREEMAIN MVS GETMAIN MVS POST Message and/or MVS ABEND Create CAVM process block.		DFHWCIPS	MVS abend Caller
DFHWCGNT	DFHWCGNA		Entry point list for CAVM modules above the 16-megabyte line.			

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWCSLM			DFHWCSLM consists of three modules link-edited together: DFHWCSNT DFHWMCAN DFHWMS It is loaded by DFHSIB1.			
DFHWCSNT	DFHWCSNA		Entry point list for CAVM modules below the 16-megabyte line.			
DFHWDAT	DFHWDAT	Many CAVM modules	Causes the current CAVM process to wait for specific events.	DFHWTRP	DFHWCGPS DFHWDGPS DFHWXBPS	DFHWDISP
DFHWDATT	DFHWDATT	DFHWDINA DFHWMG1 DFHWMP1 DFHWMR1 DFHWSXPI	Creates the CAVM process.	DFHWCCS DFHWTRP	DFHWCGPS DFHWDGPS DFHWXBPS	
DFHWDINA	DFHWDINA	DFHWSRTR	Attaches the initial CAVM process. Sets up lock tables, the dispatcher control area, the LIFO control area, and the dispatcher ESPIE, and ESTAE exits.	DFHWCCS DFHWDATT DFHWDSRP	DFHWCGPS DFHWDGPS DFHWXBPS	DFHWDISP
DFHWDISP	DFHWDISP DFHWDIND	DFHWDAT DFHWDINA	CAVM process dispatcher. Dispatches the next ready CAVM process, or waits for an external event. Dispatches the initial CAVM process.	DFHWTRP	DFHWCGPS DFHWDGPS	Dispatched process Caller of DFHWDINA
DFHWDSRP	DFHWDSRP	DFHWDINA CAVM process check/ abend	Establishes the CAVM process ESPIE/ESTAE. Performs CAVM process error handling for processes with ESPIE or ESTAE routines.	DFHWCCS	DFHWCGPS DFHWDGPS DFHWNFPS DFHWSXPS	
DFHWKP	DFHWKPNA	DFHSTP	DFHWKP takes a warm keypoint at the normal termination of CICS. This program is part of the restart component.	DFHLFA DFHZCQ	*DFHAIDDS *DFHCSADS *DFHDCTDS *DFHFCTDS *DFHICEDS *DFHJCA *DFHJCTDS *DFHJCTTE DFHKPDDS *DFHKPPDS *DFHLFM *DFHMRCDS DFHMTTWA *DFHSIT *DFHTCA *DFHTCTFX *DFHTCTLE DFHTDSDS *DFHTSBDS *DFHTSMDS DFHURDDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWLFRE	DFHWLFRE	Many CAVM modules	Frees the LIFO stack entry for CAVM modules running above the 16-megabyte line.	DFHWTRP	DFHWXBPS DFHWXLPS	
DFHWLGET		Many CAVM modules	Gets the LIFO stack entry for CAVM modules running above the 16-megabyte line.	DFHWTRP	DFHWXBPS DFHWXLPA	
DFHWMCAN	DFHWMCAN	DFHKCP	Handles the cancellation of CICS tasks that are waiting for completion of CAVM message services. Tells DFHKCP that the task is to continue after ensuring that the event is, or will be, posted.		DFHWMQPS	
DFHWMG1	DFHWMG1	DFHWDISP DFHWDSRP DFHWMI	Main module of the CAVM message manager GET MESSAGE services. Called by WMI to initialize service and attach itself as a message-reader CAVM process. Called by WDISP to run as message-reader CAVM process. Reads messages and stores them. Called by WDSRP to handle ESPIE/ESTAE exits for the message reader.	DFHWCCS DFHWDATT DFHWMQH DFHWMRD	DFHWDGPS DFHWMGPS	
DFHWMI	DFHWMI	DFHWSXPI	Allocates the CAVM message-manager communications area. Calls each of the main message-manager modules to initialize themselves.	DFHWCCS DFHWMG1 DFHWMP1 DFHWMR1	DFHWDGPS DFHWMGPS	
DFHWMMT	DFHWMMT	DFHWMRD DFHWMWR	Provides VSAM GET/PUT services for the CAVM message data set.	DFHWCCS DFHWTRP	DFHWFGPS DFHWMGPS DFHWMTPS	
DFHWMPG	DFHWMPG	DFHWMP1 DFHWMR1	Copies message data into the buffer provided by the user of PUTMSG, PUTREQ, and PUTRSP CAVM message-manager services. Provides an ESPIE routine to handle program checks occurring during the copying.		DFHWMGPS DFHWMQPS	
DFHWMP1	DFHWMP1	DFHWDISP DFHWDSRP DFHWMI	Main module of CAVM message-manager PUT MESSAGE service. Called by DFHWMI to initialize service, and attach itself as message-writer CAVM process. Called by DFHWDISP to run as message-writer CAVM process. Writes messages to the CAVM message data set. Called by DFHWDSRP to handle ESPIE/ESTAE exits for message writer. DFHWMQG	DFHWCCS DFHWDATT DFHWDWAT DFHWMQS DFHWMWR	DFHWMGPS DFHWMQPS DFHWSAPS	
DFHWMQG	DFHWMQG	DFHWMS20	Runs under the CICS TCB above the 16-megabyte line. Processes GETMSG CAVM message-manager requests. Waits for a message to arrive, then copies from main-memory message queue created by CAVM message-reader process.	DFHKCP DFHSCP DFHWCCS DFHWMQH	DFHTCA DFHWCSPS DFHWMGPS DFHWMMPS DFHWMQPS DFHWMSPS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWMQH	DFHWMQH	DFHWMG1 DFHWMQG	CAVM message-manager message input queue handler. Locates/creates message-queue anchor blocks. Adds copies of messages read by the CAVM reader process to the main-memory message queues.	DFHWCCS DFHWTRP DFHXRA	DFHWMGPS DFHWMMPS DFHWNFPS DFHWSXPS	
DFHWMQP	DFHWMQP	DFHWMS20	Runs under the CICS TCB above the 16-megabyte line. Processes CAVM message-manager PUTMSG, PUTREQ, and PUTRSP requests. Places the request in the appropriate queue. Posts the queue to awaken CAVM process to handle request, waits for completion, and returns response to the caller.	DFHKCP DFHSCP DFHWCCS	DFHTCA DFHWCSPS DFHWMGPS DFHWMQPS DFHWMSPS	
DFHWMQS	DFHWMQS	DFHWMP1 DFHWMR1	CAVM message-manager message output queue handler. Provides services to select next work item to process, and posts items complete.	DFHWCCS	DFHWMGPS DFHWMQPS	
DFHWMRD	DFHWMRD	DFHWMG1	CAVM message-manager message read routine. Reads messages from CAVM message data set, taking account of the position of the active write cursor. Creates message blocks for copies of messages read.	DFHWCCS DFHWDWAT DFHWMMT	DFHWFGPS DFHWMGPS DFHWMMPS DFHWWRPS DFHWNFPS DFHWSAPS DFHWSXPS	
DFHWMS	DFHWMSNA	Users of CAVM message services	CAVM message-manager service interface routine. Runs under the CICS TCB below the 16-megabyte line. Switches to 31-bit mode, and passes the request to DFHWMS10. Switches back to the 24-bit mode, and returns the result to caller.	DFHWMS10	DFHCSADS DFHTCA DFHWCGPS DFHWCSPS DFHWMQPS	
DFHWMS10	DFHWMS10	DFHWMS	CAVM message manager services interface. Runs under the CICS TCB above the 16-megabyte line. Builds a dummy CAVM process block, so that subsequent modules can run in an XRF LIFO environment. Calls DFHWMS20 to process request passed by the caller.	DFHWCCS DFHWMS20	DFHCSADS DFHTCA DFHWMGPS DFHWMQPS	
DFHWMS20	DFHWMS20	DFHWMS10	CAVM message manager services interface. Selects request type and passes request to DFHWMQP (PUTMSG, PUTREQ, PUTRSP) or DFHWMQG (GETMSG).	DFHWMQG DFHWMQP		
DFHWMWR	DFHWMWR	DFHWMP1	CAVM message-manager message write routine. Takes data from PUTMSG requests and copies them into CI buffers to be written to the CAVM message data sets.	DFHWCCS DFHWDWAT DFHWMMT DFHWMQS DFHWTRP	DFHWFGPS DFHWMGPS DFHWMQPS DFHWMTPS DFHWSAPS	
DFHWOS	DFHWOS		Overseer bootstrap module. Loads DFHWOSA and passes control to it.	DFHWOSA		

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWOSA	DFHWOSA	DFHWOS	Overseer services initialization module. Processes control parameters. Loads DFHWOSB and sets up entry points for overseer services.	Overseer program		
DFHWOSB	DFHWOSB	Overseer program	Overseer service. Processes requests from the overseer program that are issued by the DFHWOSM macro.			
DFHWSRTR	DFHWSMNA	DFHXRA MVS after attach of new TCB	CAVM state-management request router and subtask entry point. Initial entry point for CAVM task attached by DFHWSSN1 to process the CAVM SIGNON command. Calls DFHWSSN2 to continue the processing of the SIGNON request, and, if it is accepted, calls DFHWDINA to ATTACH the tick generator module DFHWSTI as:2 the first and highest priority CAVM process. Called under the CICS TCB to queue the CAVM TAKEOVER command for processing by the CAVM task. Called under the CICS TCB to initiate processing of the CAVM SIGNOFF command by detaching the CAVM task. Initial entry point for MVS subtasks attached by the CAVM task to perform various functions, such as issuing requests for CSVC services, or formatting new CAVM data sets when they are used for the first time.	DFHKCP DFHLFM DFHWDINA DFHWSSN2	DFHCSADS DFHTCA DFHWCSDS DFHWDSDS DFHWSTDS DFHWS2DS	
DFHWSSN1	DFHWSSNA	DFHXRA	CAVM state management SIGNON initial entry point. The CICS task issues an OS LINK, specifying load module DFHWSSON to perform a CAVM SIGNON request. DFHWSSN1 attaches the CAVM task to execute the request, waits to see if it is successful, detaches the task if not, and reports the result to CICS.	Some MVS services	DFHCSADS DFHTRACE DFHWCSDS DFHWSSDS DFHWS2DS	
DFHWSSN2	DFHWSSN2	DFHWSRTR	CAVM state management SIGNON request handler. Entered under the CAVM TCB to process a CAVM SIGNON request. Allocates storage for, and initializes, key CAVM control blocks, sets up DFHWSSOF as an ESTAE exit, calls DFHWSSN3 to OPEN the CAVM data sets, reads the state management record from the control data set, uses the JES inquire-job-status CSVC service provided by DFHWTI, and looks for surveillance signals from other CAVM users to check whether the environment is such that the requested SIGNON can be accepted. Prompts the operator for job status information if necessary. If SIGNON is accepted, updates the state management record and status CIs, to record that this job has signed on to the CAVM. When possible, also cleans up out-of-date information in the CAVM data sets left behind by jobs which were unable to sign off properly before terminating.	DFHCSVC DFHWCCS DFHWSSN3 DFHXRC Many MVS and DFP services	DFHAFCD DFHWCGPS DFHWCSDS DFHWDSDS DFHWFGDS DFHWSADS DFHWSMDS DFHWSRDS DFHWSSDS DFHWSXDS DFHWS2DS DFHWS3DS DFHWTADS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWSSN3	DFHWSSN3	DFHWSSN2	CAVM state management data set initialization routine. Builds ACBs, opens and validates the CAVM control and message data sets for CAVM SIGNON. Builds reserve parameter list for serializing accesses to the control data set. If new CAVM data sets are being used for the first time, attaches an MVS subtask to record pertinent information in each data set's control interval, and to format the CIs needed by state management.	Many MVS and DFP services	DFHWFGDS DFHWS3DS	
DFHWSSOF	DFHWSSOF	MVS recovery/ termination manager	CAVM state management SIGNOFF request handler. During SIGNON processing, this module is established as an ESTAE exit for CAVM task. Purges outstanding I/O requests, reads the state management record from the control data set, and searches it to see if this job has signed on to the CAVM. If so, updates the status CI and state management record to indicate that job has signed off. Makes the TAKEOVER message available to DFHWSRTR when an active system signs off after takeover has started.	Many MVS and DFP services	DFHWCGDS DFHWCSDS DFHWFGDS DFHWSADS DFHWSMDS DFHWSRDS	
DFHWSSR	DFHWSSR	DFHWDISP	CAVM surveillance status reader. Runs as a process controlled by the XRF dispatcher, DFHWDISP. Reads the status CI of the partner system from the control data set or the message data set, generates internal CAVM events, and/or drives the NOTIFY exit when the partner's status changes, or its surveillance signals cease. For an alternate system, monitors, and records the time-of-day clock difference when the active system is running in a different CEC.	DFHWDWAT DFHWLGET DFHXRB Some MVS and DFP services	DFHWCGDS DFHWDSDS DFHWFGDS DFHWNFDS DFHWSADS DFHWSCDS DFHWSRDS DFHWSXDS	
DFHWSSW	DFHWSSW	DFHWDISP	CAVM surveillance status writer. Runs as a CAVM process controlled by the CAVM dispatcher, DFHWDISP. Writes a system's current status to its status CI in the control data set, or the message data set, to make it available to its partner and to provide a surveillance signal. Generates an internal CAVM event when a status write completes. Puts the current time-of-day clock reading in the status CI to permit DFHWSSR to deduce the time-of-day clock difference when the active system and the alternate system are running in different CECs.	DFHWDWAT DFHWLGET Some MVS and DFP services	DFHWCGDS DFHWDSDS DFHWFGDS DFHWSADS DFHWSRDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWSTI	DFHWSTI	DFHWDISP	CAVM surveillance tick generator and CICS status monitor. Runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. Issues an MVS STIMER for the surveillance interval, and, when this expires, generates an internal CAVM clock-tick event, calls the inquire-CICS-status exit and schedules the surveillance status writer processes, to cause a surveillance signal reporting this system's current status to be written to the control data set or the message data set.	DFHWDWAT DFHWLGET DFHWSXPI DFHXRC MVS POST and STIMER	DFHWCGDS DFHWCSDS DFHWDSDS DFHWSADS DFHWSNDS DFHWSXDS	
DFHWSTKV	DFHWSTKV	DFHWDISP	CAVM state management TAKEOVER request handler. Runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. When a new active SIGNON has been detected, reads the state management record from the control data set and attaches an MVS subtask to invoke DFHWTI's validate-CLT CSVC service. When a TAKEOVER command has been issued, reads the state management record, validates the TAKEOVER request and attaches an MVS subtask to use DFHWTI's JES inquire-job-status service to determine the current state of the active system. If the active system is still signed on to CAVM, updates the state management record to indicate that a takeover is in progress, places the TAKEOVER message for the active system in the alternate system's status, and attaches an MVS subtask to invoke DFHWTI's TAKEOVER-initiate service. Once the active system has signed off (or terminated), requests DFHWSSR to read the active system's final status, quiesces surveillance processing, updates the state management record and status CIs to indicate the stage reached by takeover. Then arranges for surveillance processing to be resumed in active mode. Attaches an MVS subtask to invoke DFHWTI's process-CLT CSVC service if necessary. When the active system has finally terminated, updates the state management record to take its place as the new active system. Generates internal CAVM events and/or calls the NOTIFY exit to report the progress of the TAKEOVER request, including acceptability of the time-of-day clock reading. Terminates by returning to DFHWDISP.	DFHCSVC DFHWDWAT DFHWLFRE DFHWLGET DFHXRB Many MVS and DFP services	DFHAFCD DFHWCGDS DFHWCSDS DFHWDSDS DFHWFGDS DFHWNFDS DFHWSADS DFHWSCDS DFHWSMDS DFHWSTDS DFHWSXDS DFHWTADS	
DFHWSXPI	DFHWSXPI	DFHWSTI	CAVM state management CAVM process initialization. Runs under the tick generator CAVM process towards the end of SIGNON. Attaches the TAKEOVER CAVM process (alternate systems only), two status writer CAVM processes, two status reader CAVM processes, and then calls the CAVM message management initialization module.	DFHWDATT DFHWLFRE DFHWLGET DFHWMI MVS GETMAIN	DFHWCGDS DFHWCSDS DFHWDSDS DFHWSADS DFHWSCDS DFHWSNDS DFHWSTDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHWTI	DFHWTINA	DFHCSVC from: DFHWSSN2 DFHWSTKV DFHZXSTS	Takeover initiation is the primary function of this module, and is requested by CAVM state management at takeover to terminate the CICS active system issue commands in the CLT, and wait until the CICS active system terminates. Other XRF services provided by this module are to determine whether a job is running, to issue the operator commands for the overseer program, to issue MODIFY USERVAR to VTAM, to validate the CLT, and to process the CLT.		DFHAFCD DFHAFCS DFHCLTDS DFHSABDS DFHWTADS	MVS to: DFHWSSN2 DFHWSTKV DFHZXSTS
DFHWTRP		Many CAVM modules	Makes a trace entry in CAVM main-memory trace table.	DFHWCCS	DFHWTGPS DFHWTRPS DFHWXBPS	
DFHXFP	DFHXFPNA	DFHISP DFHMIR	The online data transformation program takes data addressed from a parameter list (command-level or DL/I), and constructs an FMH suitable for transmission to a remote ISC system; DFHXFP also performs the reverse transformation.	DFHPCP DFHSCP	*DFHCSADS *DFHDCTDS *DFHDRX *DFHEISDS *DFHFCTDS DFHFMHDS *DFHPCTDS *DFHSEC DFHTCADS *DFHTCTTE DFHTIOA *DFHTSHD DFHTSTDS *DFHXFSTG	
DFHXFQ	DFHXFQNA	DFHDRPE	The batch transformation program executes in a shared data base region. DFHXFQ takes data addressed from a DL/I parameter list and constructs an FMH suitable for passing to the online region; DFHXFQ also performs the reverse transformation.	None	*DFHXFMOD	
DFHXFX	DFHXFXNA	DFHISP DFHMIR	DFHXFX performs the same logical transformations of function shipping requests as DFHXFP but in a manner that is optimized for the MRO environment. It is not used for the transformation of DL/I requests.	DFHPCP DFHSCP	*DFHCSADS *DFHDCTDS *DFHEIFD *DFHEIS DFHFCENT *DFHFCTDS *DFHPCTDS DFHTCA *DFHTCTTE DFHTIOA *DFHTSHD DFHTSTDS *DFHXFSTG	
DFHXJCC	Defined by user	DFHJCC	The user-replaceable module, DFHXJCC, is called by DFHJCC during close processing. It receives control just after the close request was completed by the journal subtask.	DFHJCC	*DFHCSADS *DFHJCA *DFHJCTTE *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHXJCO	Defined by user	DFHJCO	The user-replaceable module, DFHXJCO, is called by DFHJCO during open processing. It receives control just before the open request is passed to the journal subtask.	DFHJCO	*DFHCSADS *DFHJCA *DFHJCTTE *DFHTCA	
DFHXMP	DFHXMPNA	DFHDRPE DFHZCX	The cross-memory program is invoked by a program call (PC) instruction and uses MVS cross-memory services to pass data from one subsystem to another within the same processing unit. The communicating subsystems are usually in CICS address spaces, but DFHXMP does not assume this.	None	*DFHAFCD *DFHIRPD *DFHIRSDS	
DFHXRA	DFHXRANA	DFHJCSDJ DFHSIC1 DFHSII1 DFHSRP DFHSTP DFHTCRP DFHTDRP DFHWMQH DFHWMR1 DFHXRCP DFHXRSP DFHZNAC DFHZOPN DFHZSIS	DFHXRA is the program that executes the DFHXR macro. It runs under the CICS TCB in AMODE(24). In general, it uses CICS macros to invoke other services. Exceptions are OS LINK to DFHWSSON to sign on to the CAVM, and OS LOAD and DELETE for DFHWSMS to sign off from the CAVM, and to initiate takeover. It invokes global user exit XXRSTAT, which can lead to the abend 208.	DFHKCP DFHUEH DFHWSMS DFHWSSON	DFHCSADS DFHTCA DFHXRSPS DFHWSSPS	
DFHXRB	DFHXRBNA	DFHWDSRP DFHWMQH DFHWMRD DFHWSSR DFHWSTKV	XRF notify exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31). It reacts to events detected by various CAVM modules. It creates a queue of work elements (chained from XRWECHN) to be processed by DFHXRSP.		DFHWNFPS DFHXRHPS DFHXRSPS DFHXRWPS	
DFHXRC	DFHXRCNA	DFHWSSN2 DFHWSTI	CICS-status exit program. The address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31). It returns the latest CICS-status data to be written to the state management data set.		DFHXRHPS DFHXRSPS	
DFHXRCP	DFHXRCNA	DFHXRF	XRF console communication task runs under the CICS TCB in AMODE(24). It processes modify commands received by CICS during initialization of the alternate system. It initiates takeover, shuts down the active system, and manages trace and dump as required.	DFHFDP DFHICP DFHKCP DFHOCP DFHSGP DFHTRP DFHXRA	DFHCSADS DFHTCA DFHXRSPS	
DFHXRD			XRF bootstrap program for the CAVM system transaction (SWDT). DFHXRD is attached during initialization, and links to DFHXRSP.	DFHPCP	*DFHCSAPS *DFHTCAPS DFHXRSPS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHXRE	DFHXRENA	DFHPCP	XRF bootstrap program 2. The entry point for the system task attached by DFHXRA. The system task links to DFHXRCP, the XRF console communication program.	DFHXRCP	DFHCSADS DFHTCADS DFHXRSPS	
DFHXRF	DFHXRFNA	DFHPCP	XRF bootstrap program 3. This is the entry point for the system task attached by DFHXRA. The system links to DFHXTCI, the XRF terminal switch scan program.	DFHXTCI	DFHCSADS DFHTCA DFHXRSPS	
DFHXRP			DFHXRP consists of five object modules link-edited together: DFHXRA DFHXRB DFHXRC DFHXRE DFHXRF It is loaded by DFHSIB1.			
DFHXRSP	DFHXRSNA	DFHXRA	XRF surveillance program, loaded by DFHSIB1, but runs as a program under a CICS transaction. It runs under the CICS TCB in AMODE(24). It processes the queue of work elements created by DFHXRB. It attaches the catch-up transaction CXCU, initiates takeover, and shuts down CICS as required. It can issue abends 206 and 207.	DFHICP DFHKCP DFHSTP DFHTRP DFHXRA	DFHCSADS DFHTCA DFHXRSPS DFHXRWPS	
DFHXSE	DFHXSENA		This module performs external security checks. It handles resource checking, sign-on with password, sign-on without password, and initialization of the security manager.	DFHPCP DFHSCP DFHSKP	*DFHAFCD *DFHCSADS *DFHSECDS *DFHSIT *DFHSNT *DFHTCA *DFHTCTFX *DFHTCTLE *DFHTCTZE	
DFHXSP	DFHXSPNA	DFHDLI DFHEDF DFHSFP DFHSIG1 DFHSNP DFHZSUP	The security program provides the interface to a security manager and provides standard CICS transaction verification and resource-level check. DFHXSP creates and frees a pseudo SNT entry for each sign-on.	DFHXSE	*DFHAFCD *DFHCSADS *DFHLFM *DFHPCTDS *DFHSECDS *DFHTCA *DFHTCTZE	
DFHXSS	DFHXSPNA	DFHCSVC	This module provides the interface to RACF, using RACROUTE macro calls.	RACF	*DFHACEE *DFHCSADS *DFHSECDS	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHXTCI	DFHXTCI		Transaction invoked when the alternate system begins a takeover. It examines the TCT to locate the terminals with XRF backup sessions. It queues these TCTTEs to DFHZSES for the SessionC Control=Switch command.		DFHTCTFE DFHTCTTE	
DFHXTP	DFHXTPNA	DFHTPS DFHZTSP	The terminal sharing transformation program comprises four logical modules (known as transformers 1 to 4). DFHXTP transforms routing requests into the logical unit type 6 format for shipping to a remote CICS address space.	DFHPCP DFHSCP DFHZLOC	*DFHCSADS DFHFMHDS *DFHLFM *DFHMCRDS DFHSNNT *DFHTCA *DFHTCTZE DFHTIOA DFHURLDS *DFHXTSTG	
DFHZABD	DFHZABD1	TC CTYPE= requests	If a TC CTYPE request is issued when ZCP been generated without VTAM support, then DFHZABD is invoked to abend the transaction.	DFHPCP	None	
DFHZACT	DFHZACT1	DFHZDSP	The activate scan routine scans the four TCTTE activity queues - activate, log, wait, and NACP. DFHZACT scans the activate queue for request bits that may be set in the TCTTEs; for each request, DFHZACT calls the appropriate module. If no requests are outstanding, the TCTTE is removed from the queue. If the NACP queue is not empty, DFHZACT attaches DFHZNAC (if not already attached). Similarly, if the log queue is not empty, DFHZACT attaches DFHZRLG. DFHZACT scans the wait queue. If automatic resource definition is in the system, DFHZACT looks for any corresponding work elements. For each work element, DFHZATD is attached.	DFHKCP DFHZATD DFHZATI DFHZATT DFHZCLS DFHZDET DFHZFRE DFHZGET DFHZOPN DFHZQUE DFHZRST DFHZRSY DFHZRVS DFHZSDA DFHZSDR DFHZSDS DFHZSES DFHZSIM DFHZSKR DFHZTRA	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZAIT	DFHZAIT1	DFHSIF1	The attach initialization tables routine initializes local tables used by the mainline task attach routine, DFHZATT. DFHZAIT generates the page command table from information supplied by the system initialization table and modifies DFHZATT appropriately. DFHZAIT initializes the transaction code delimiter table and the AID transaction code table.	None	*DFHCSADS *DFHSIT	
DFHZAND	DFHZAND1	DFHZARQ	The abend control block builder is used to assist in building the transaction abend block when an abend has occurred in an interconnected system. Its function is to extract the error sense bytes, and diagnostic message sent by the other system and to copy these into the block. As an initial step in its processing, DFHZAND acquires storage for the block itself.	None	*DFHTCADS *DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZARL	DFHZARL1	DFHEGL DFHETL DFHSPP DFHZARM DFHZLUS DFHZSUP	DFHZARL interprets all LU6.2 requests. For SEND requests, it calls DFHZSDL, for RECEIVE requests DFHZRVL. If the request and the data format are valid, DFHZARL updates the send buffer and the current state of the session.	DFHKCP DFHLFO DFHSCP DFHZISP DFHZRVL DFHZSDL	DFHCSADS DFHLUCDS DFHTCADS DFHTCTTE	
DFHZARM	DFHZARM1	DFHZARQ DFHETL DFHSPP	DFHZARM handles DFHTC macros for LU6.2 sessions.	DFHLFO DFHZARL	DFHCSADS DFHTCADS DFHTCTTE	
DFHZARQ	DFHZARQ1	DFHISP EIP trans TC requests	The application request interface module analyzes the DFHTC macro request from the application. For a VTAM terminal, it sets the appropriate flags and calls the module required or adds the TCTTE to the activate chain. For a BTAM terminal, the request is copied to the TCTTE and left for DFHTCP.	DFHJCP DFHKCP DFHPCP DFHZARM DFHZFRE DFHZRAQ DFHZRAR DFHZSDS	*DFHCSADS *DFHJCADS *DFHTCADS *DFHTCTTE DFHTIOA	
DFHZASX	DFHZASX1	VTAM	The asynchronous command exit module is called by VTAM if an asynchronous command is received. The only such commands are: request shutdown, quiesce at end of chain, release quiesce and signal. DFHZASX sets up the TCTTE appropriately and returns control to VTAM.	None	*DFHCSADS DFHTCTFX *DFHTCTTE	
DFHZATD	DFHZATD	DFHZACT	The autoinstallation program, DFHZATD, performs functions relating to the INSTALL and DELETE process. It requests information from a user program where appropriate.	DFHZCQ	*DFHCSADS *DFHTCA DFHTCPZR *DFHTCTFX *DFHTCTTE *DFHTCTZE DFHZCQPS DFHZEPD IFGRPL	
DFHZATDX	DFHZATDX	DFHZATD	DFHZATDX is the user program for autoinstall. It is called when INSTALL is in progress or DELETE has completed. For INSTALL, DFHZATDX selects a model name, and the corresponding TRMIDNT, to be used by the terminal control builder program (DFHTBSxx). This program can be used as a model for a user program.			
DFHZATI	DFHZATI1	DFHZACT	The automatic task initiation module checks for stress conditions, calls DFHZSIM if the node is not in session, acquires an RPL if necessary, issues a conditional DFHKC TYPE=AVAIL macro. DFHZATI initiates bid protocols to decide if LU is available or not.	DFHKCP DFHZGET	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZATT	DFHZATT1	DFHZACT	The task attach module checks for stress conditions, allocates an RPL if necessary, determines the task to be attached either from the data, from the TCTTE (if the previous transaction specified TRANID), or from the AID (for a 3270). DFHZATT also checks for paging commands (having been modified by DFHZAIT). Finally a conditional ATTACH is issued. The module is applicable for VTAM, SRL and OS console support.	DFHKCP DFHSCP DFHZGET DFHZSDS	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZBAN	DFHZBAN	DFHZOPN	The terminal control bind analysis program checks that a bind is valid and supportable and, if requested, sets the TCTTE information that supports the session parameters.		*DFHCSADS *DFHTCA	
DFHZBKT	DFHZBKT1	DFHZSDL DFHZSLX DFHZRLX DFHZLUS	DFHZBKT maintains the bracket state for LU6.2.	None	DFHTCTTE	
DFHZCA	DFHZCANA	None	DFHZCA is the generic name allocated to a composite module consisting of a set of ZCP copy books. It is not called by anyone. DFHZCA consists of the following modules: DFHZACT - Activate chain DFHZFRE - FREEMAIN request DFHZGET - GETMAIN request DFHZQUE - Chaining DFHZRST - RESETSR.	None	None	None
DFHZCB	DFHZCBNA	None	DFHZCB is the generic name allocated to a composite module consisting of a set ZCP copy books. It is not called by anyone. DFHZCB consists of the following modules: DFHZATI - Automatic task initiation DFHZDET - Task detach DFHZLRP - Logical record presentation DFHZRAC - Receive any completion DFHZRVS - Receive specific DFHZRVX - Receive specific exit DFHZSDR - Send response DFHZSDS - Send DFSYN DFHZSDX - Send DFSYN data exit DFHZSSX - Send DFSYN exit DFHZUIX - User input exit	None	None	None

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZCC	None	None	DFHZCC is the generic name allocated to a composite module consisting of a set of ZCP copy books. It is not called by anyone. DFHZCC consists of the following modules: DFHZARL - LU6.2 request process program DFHZARM - LU6.2 macro requests program DFHZBKT - LU6.2 bracket state program DFHZCHS - LU6.2 chain state program DFHZCNT - LU6.2 contention state program DFHZRLX - LU6.2 receive exit program DFHZRVL - LU6.2 receive program DFHZSDL - LU6.2 send program DFHZSLX - LU6.2 send exit program DFHZUSR - LU6.2 conversation state program.	None	None	None
DFHZCHS	DFHZCHS1	DFHZRLX DFHZSDL DFHZSLX	DFHZCHS maintains the chain state for LU6.2.	None	DFHTCTTE	
DFHZCLS	DFHZCLS1	DFHZACT	The close destination module obtains an RPL if necessary, issues CLSDST to VTAM, and checks if it was accepted. The CLSDST exit handles the completion of the request. DFHZCLS performs a normal closedown procedure according to the LU type (for example, LU6 sends SBI and BIS). In the case of an abnormal closedown, DFHZCLS performs immediate termination, using CLSDST or TERMSESS commands. If the terminal was automatically defined, it is put out of service.	DFHZGET	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZCLX	DFHZCLX1	VTAM	The close destination exit module receives control from VTAM when a CLSDST, or TERMSESS request completes. If the CLSDST or TERMSESS was successful, DFHZCLX cleans up TCTTE and returns to VTAM; otherwise, it enqueues the TCTTE to DFHZNAC, and then returns to VTAM.	None	DFHTCTFX *DFHTCTTE	
DFHZCNA	DFHZCNA1	DFHZDSP	The system console activity control program is responsible for CICS system requests. It performs the following functions: Shutdown - When all other access method terminals have been quiesced, then console support is quiesced, allowing CICS to terminate. Resume - Resumes tasks waiting on read request when they are completed. Detach - Releases all TIOAs associated with a completed task. Attach - Passes the data associated with a MODIFY command (in a TIOA attached to a console TCTTE) to DFHZATT to create a task. ATI - Determines if a console TCTTE is available for automatic task initiation.	DFHKCP DFHSCP DFHZATT DFHZSUP	DFHTCTCE DFHTCTFX *DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZCNR	DFHZCNR1	DFHZARQ	The system console application request program performs READ and WRITE operations to an OS/VS system console which is used as a terminal.	DFHSCP	DFHTCTCE *DFHTCTTE DFHTIOA	
DFHZCNT	DFHZCNT1	DFHZRLX DFHZLUS	DFHZCNT maintains the contention state for LU6.2.	None	DFHTCTTE	
DFHZCP	DFHZCPNA	None	DFHZCP is the generic name allocated to a composite module consisting of a set of ZCP copy books. It is not called by anyone. DFHZCP consists of the following modules: DFHZARQ - Application request DFHZATT - Attach routine DFHZCNA - System console activity control program DFHZDSP - Dispatcher DFHZISP - Allocate free routine DFHZSUP - Startup task. DFHZUCT - 3270 uppercase translation.	None	None	None
DFHZCQ	DFHZCQ	DFHAMP DFHQRY DFHTCRP DFHTOR DFHWKP DFHZATD DFHZTSP	DFHZCQ is the control program for all requests for the dynamic add and delete of terminal control table entries. It is called by resource definition online (RDO) to: COLD start group lists COLD/WARM start nonmigrated VTAM resources Dynamically install using the CEDA transaction The main subroutines of DFHZCQ are: DFHZCQIS Install DFHZCQIT Install TCTTE DFHZCQIN Initialize DFHZCQDL Delete DFHZCQRS Restore DFHZCQRC Recover DFHZBAN Validate bind	DFHBSxx DFHLFA DFHSCP	*DFHCSADS *DFHLFM *DFHTCA	
DFHZCQDL	DFHZCQDL	DFHZCQOO DFHZNAC RDO	DFHZCQDL dynamically deletes a TCT entry when the entry is quiesced. This module is part of DFHZCQ.	DFHLFA DFHSCP	*DFHCSADS *DFHLFM *DFHTCA	
DFHZCQIN	DFHZCQIN	DFHTCRP	DFHZCQIN initializes DFHZCQ for all its operations. This module is part of DFHZCQ.	DFHLFA DFHTMP	*DFHCSADS *DFHLFM *DFHSIT *DFHTCA *DFHTCTFX *DFHTCTTE	
DFHZCQIQ	DFHZCQIQ	DFHZTSP	DFHZCQIQ obtains the parameters for a TCT resource and is called by DFHZTSP in the terminal-owning node as part of the process of shipping a TCT definition to a remote system. This module is part of DFHZCQ.	DFHLFA DFHSCP	*DFHCSADS *DFHLFM *DFHTCA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZCQIS	DFHZCQIS		DFHZCQIS installs a TCTTE. If the resource already exists, the old resource is deleted.	DFHLFA DFHSCP	*DFHCSADS *DFHLFM *DFHTCA DFHZCQGB	
DFHZCQIT	DFHZCQIT		DFHZCQIT adds a macro-generated TCTTE to a CICS system.	DFHLFA DFHSCP DFHTMP	*DFHCSADS *DFHLFM *DFHTCA *DFHTCT *DFHTCTCE *DFHTCTSK *DFHTCTTE DFHTMRQ	
DFHZCQRC	DFHZCQRC		DFHZCQRC merges records from the recovery control log. The records are the terminal control installs and/or deletes that were being committed at the time of a system crash.		*DFHCSADS *DFHTCA	
DFHZCQRS	DFHZCQRS		During emergency restart or warm start, DFHTCRP restores terminal control resources to the state they were in before the last shutdown of CICS, using the restart data set.	DFHLFA DFHSCP	*DFHCSADS *DFHLFM *DFHTCA DFHZCQGB	
DFHZCRQ	DFHZCRQ1	TC CTYPE requests	The CTYPE request module analyzes DFHTC CTYPE commands, and calls or links to the appropriate send module.	DFHKCP	*DFHCSADS *DFHTCTTE	
DFHZCW	None	None	DFHZCW is the generic name allocated to a composite module consisting of a set of ZCW copy books. It is not called by anyone. DFHZCW consists of the following modules: DFHZERH LU6.2 error program DFHZLUS LU6.2 session management program.	None	None	Not applic
DFHZCX	DFHZCXNA	None	DFHZCX is the generic name allocated to a composite module consisting of a set of ZCP copy books. It is not called by anyone. DFHZCX consists of the following modules if ISC: DFHZABD - Abend routine for invalid requests DFHZAND - Build TACB before issuing PC abends DFHZCNR - System console application request program DFHZIS1 - CTYPE requests DFHZIS2 - IRC requests DFHZLOC - Locating TCTTE and ATI request DFHZSTU - Status changing of TCTTEs/LDCs and TCTSEs DFHZTSP - Terminal sharing.	None	None	Not applic

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZCY	DFHZCYNA	None	DFHZCY is the generic name allocated to a composite module consisting of a set of ZCY copy books. It is not called by anyone. DFHZCY consists of the following modules: DFHZASX - DFASY exit DFHZDST - SNA-ASCII translation DFHZDWE - DWE processing DFHZLEX - LERAD exit DFHZLGX - LOGON exit DFHZLTX - LOSTERM exit DFHZNSP - Network services exit DFHZOPA - Open VTAM ACB DFHZRRX - Release request exit DFHZRSY - Resynchronization DFHZSAX - Send synchronous command exit DFHZSCX - SESSION control input exit DFHZSDA - Send synchronous command DFHZSES - SESSIONC DFHZSEX - SESSIONC exit DFHZSHU - Shutdown VTAM DFHZSIM - SIMLOGON DFHZSIX - SIMLOGON exit DFHZSKR - Send response to command DFHZSLS - Set logon start DFHZSYN - Handle CTYPE=sync point/recover request DFHZSYX - SYNAD exit DFHZTPX - TPEND exit DFHZTRA - Create ZCP/VIO trace requests.	None	None	Not applic
DFHZCZ	DFHZCZNA	None	DFHZCZ is the generic name given allocated to a composite module consisting of a set of ZCP copy books. It is not called by anyone. DFHZCZ consists of the following modules: DFHZCLS - CLSDST request DFHZCLX - CLSDST exit DFHZCRQ - Command request DFHZEMW - Error message writer DFHZOPN - OPNDST DFHZOPX - OPNDST exit DFHZRAQ - Read ahead queuing DFHZRAR - Read ahead retrieval DFHZTAX - Turnaround exit.	None	None	Not applic
DFHZDET	DFHZDET1	DFHZACT DFHZISP	The task detach module receives control when a detach request is issued by DFHZISP. If a WRITE is pending (deferred write or any write), the send routine is called. If send cannot complete, the detach request is left on activate queue. If requests are queued then DFHZACT will re-drive DFHZDET when operation is complete. If the node is in between bracket state, an end bracket is sent.	DFHKCP DFHSCP DFHTSP DFHZEMW DFHZRST DFHZSDR DFHZSDS	*DFHCSADS *DFHTCADS *DFHTCTTE	
DFHZDSP	DFHZDSP1	DFHSIP	The dispatcher module handles the dispatching of modules for execution, and gives control to VTAM modules of DFHZCP using DFHZACT, or to DFHTCP (BTAM half of system) as appropriate.	DFHKCP DFHTCP DFHZACT DFHZCNA DFHZRAC DFHZSHU DFHZSLS	*DFHCRBDS *DFHCSADS *DFHIRSDS DFHTCTFX *DFHTCTTE	None

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZDST	DFHZDST1	DFHZRVX DFHZSDS	The data stream translator module translates data between EBCDIC and ASCII code as that data is being sent and received on VTAM sessions.		DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZDWE	DFHZDWE1	DFHSPP	The deferred work element module receives control from DFHSPP after DFHSPP has logged the data associated with the DWE (deferred work element). DFHZDWE establishes logical sequence numbers and checks if there is a logical or physical EOT. If logical, the DWE is unchained; if physical the TCTTE is interrogated for a deferred write pending and is placed on the response-awaited queue.	DFHJCP	*DFHDWEDS *DFHTCADS *DFHTCTTE	
DFHZEMW	DFHZEMW1	DFHACP DFHSPP DFHZDET DFHZNAC DFHZRAC	The error message writer module handles all requests for error messages VTAM supported terminals/LUs. According to the request flags, it: • Sends a negative response. • Purges unprocessed inbound data until EOC or CANCEL is received. • Sends an error message.	DFHMGP DFHSCP DFHZRST DFHZSDA DFHZSDR DFHZSDS	*DFHTCADS DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZERH	DFHZERH1	DFHZARL	DFHZERH handles LU6.2 error conditions.	DFHMGP DFHPCP DFHZARL DFHZBKT DFHZCHS DFHZRVL DFHZSDL	DFHCSADS DFHTCADS DFHTCTTE	
DFHZEV1	DFHZEV1		DFHZEV1 is the LU6.2 bind-time security encryption validation program, part 1.	DFHDES	*DFHTCA *DFHTCTTE	
DFHZEV2	DFHZEV2		DFHZEV2 is the LU6.2 bind-time security encryption validation program, part 2.	DFHDES	*DFHTCA *DFHTCTTE	
DFHZFRE	DFHZFRE1	DFHZACT	The FREEMAIN module is used to free storage (RPLs, NIBs, bind areas, and TIOAs) acquired by DFHZCP. Some storage is also freed by other DFHZCP modules.	DFHSCP	*DFHTCADS *DFHTCTTE	
DFHZGET	DFHZGET1	DFHZACT DFHZCLS DFHZRVS DFHZSDR DFHZSDS DFHZSES DFHZSKR	The GETMAIN module is used to acquire an RPL, NIB, bind area, or TIOA. DFHZGET also sets up the dynamic NIB using the information in the NIB descriptor block. Normally when a module requires storage, it issues a conditional GETMAIN itself, and only if this is unsuccessful does it invoke DFHZGET using the activate scan.	DFHACP DFHPCP DFHSCP	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZHPRX	DFHZHPNA	DFHKCSP (through ZHPSR and KCP)	In authorized path SRB mode, issues VTAM EXECRPL.	DFHKCP VTAM	*DFHCSADS DFHHTADS DFHTCTFX *DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZHPSR	DFHZHPS1	DFHZRVS DFHZSDS	This is the SEND and RECEIVE module for the HPO environment.	DFHKCP DFHSCP	*DFHCSADS DFHHTADS *DFHTCADS DFHTCTFX DFHZEPD	
DFHZISP	DFHZISP1	DFHISP DFHKCP DFHSPP	The intersystem program services ISC requests to free, or point to a particular TCTTE within a specified system or to allocate a TCTTE within a specified system. DFHZISP also handles ATI requests, and checks for a terminal time-out.	DFHALP DFHSCP DFHZALC DFHZARL	DFHSNNT *DFHTCTZE	
DFHZIS1	DFHZIS11	DFHSPP	Handles the transmissions control CTYPE requests of Prepare, Sync Point Request (SPR), Commit, and Abort. Each request is translated into the appropriate ISC/IRC action and is transmitted to the connected system.	DFHJCP DFHKCP DFHSCP DFHZEMW DFHZRAQ DFHZRAR	*DFHCSADS *DFHJCADS *DFHPCTDS *DFHTCADS *DFHTCTTE DFHTIOA	
DFHZIS2	DFHZIS21	DFHZARQ DFHZIS1	This module handles IRC requests for RECEIVE, DISCONNECT, IORC interregion I/O), GETDATA (fetch interregion input into TIOA), STOP, LOGOFF, OPERATIVE (IN SERVICE), RECEIVE ABORT.	DFHIRP DFHKCP DFHSCP	*DFHCRBDS *DFHIRSDS	
DFHZLEX	DFHZLEX	VTAM	The logical error address (LERAD) exit module receives control from VTAM when a logical error is detected. Logical errors are usually the result of an incorrectly defined terminal table.	None	*DFHTCTTE	
DFHZLGX	DFHZLGX1	VTAM	The logon exit module receives control from VTAM when a terminal logs on to the network. DFHZLGX scans the CICS NIBs and, if a match is found, sets an OPNDST request in the corresponding TCTTE and places it on the activate queue. If no match is found, DFHZLGX defines a terminal automatically, if possible, by allocating an autodefine work element which holds the CINIT_RU. The work element is then queued for activate scan processing. Otherwise, a dummy TCTTE is placed on the NACP queue to write an error message.	None	*DFHCSADS DFHTCTFX *DFHTCTTE DFHTCTWE	
DFHZLOC	DFHZLOC1	DFHKCP DFHZLUS	The locate module provides two functions: • Locates specific TCTTEs in the TCT. • Locates mode names.	None	*DFHCSADS *DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZLRP	DFHZLRP1	DFHZRVS DFHZSUP	The logical record presentation module handles deblocking of input data. The delimiters which are recognized are new line (NL), interchange record separator (IRS), and transparent (TRN). One logical record is returned for each DFHTC TYPE=READ request.	DFHSCP	*DFHTCADS *DFHTCTTE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZLTX	DFHZLTX1	VTAM	The lost terminal (LOSTERM) exit module receives control when VTAM detects a loss of contact with a node. There are three possible return codes set by VTAM on entry to this routine: • On return code of "node lost, recovery in progress", the terminal is placed out of service with no further action taken. • On return code of "node lost, recovery successful", the TCTTE is queued to the NACP queue with a "successful" error code set. NACP issues a CLSDST, schedules a SIMLOGON, and issues an information message. • On return code of "node lost, no recovery/or unsuccessful recovery", an "unsuccessful" error code is set and the TCTTE is queued to the NACP queue. NACP issues a CLSDST and issues the appropriate message.	None	*DFHTCTTE	
DFHZLUS	DFHZLUS1		DFHZLUS handles session management for LU6.2 sessions.	DFHMGP DFHSCP DFHZARL DFHZLOC	DFHLUSDS DFHTCADS DFHTCTME DFHTCTSE DFHTCTTE	
DFHZNAC	DFHZNANA	DFHZCP	The network abnormal condition program is attached by DFHZCP when an error in communication with a logical unit occurs. DFHZNAC performs the following functions: • Analyzes abnormal conditions. • Sends appropriate messages to the CSMT transient data destination (for terminal errors) or to the CSTL transient data destination (for logical errors). • Invokes the user-supplied (or sample) node error program. • Takes the appropriate actions resulting from the defaults which may have been modified by the node error program. • Deletes the terminal definition if it was automatic. DFHZNAC consists of the following copy books: DFHZNCA - Primary error action table and exits DFHZNCE - Take action routine DFHZNCS - Sense decode routine.	DFHJCP DFHKCP DFHMGP DFHPCP DFHTDP DFHTDP DFHZARL	*DFHDWEDS DFHIMSDS DFHTDOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZNEP	DFHZNENA	DFHZNCE (in DFHZNAC)	The CICS supplied sample node error program assists the user by providing: • A general environment within which it is easy for users to add their own error processors • Fundamental error recovery actions for a VTAM 3270 network • The default NEP where the user selects an NEP at system initialization.	None	*DFHTCTZE	
DFHZNSP	DFHZNSP1	VTAM	The network service program is invoked when VTAM detects a network service error, for example when attempting to connect two nodes together or when the link between two nodes is broken unexpectedly. This module receives control from the VTAM NSEXIT.	None	*DFHTCTTE	
DFHZOPA	DFHZOPA1	DFHMTPD	The open VTAM ACB module is invoked by DFHMTPD when the master terminal command VTAM OPEN is issued. The ACB is opened and DFHZSLS is called to accept logon requests.	DFHZSLS	*DFHCSADS DFHTCTFX	
DFHZOPN	DFHZOPN1	DFHZACT	The open destination module sets appropriate GETMAIN flags to acquire an RPL and NIB and bind areas if the TCTTE does not have these resources already, and sets up the bind image if required. DFHZOPN then issues a VTAM OPNDST or OPNSEC macro if secondary, to respond to the bind and to establish a session between CICS and the terminal.	DFHDES DFHZGET	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZOPX	DFHZOPX1	VTAM	The open destination exit module receives control from VTAM on completion of the OPNDST macro in DFHZOPN. If the OPNDST was successful, it indicates in the TCTTE that SDT (start data transfer) is to be sent and checks whether a "good morning" message should be triggered. It then returns to VTAM.	None	*DFHCSADS DFHTCTFX *DFHTCTTE	
DFHZQUE	QUEI	All ZCP exits called by VTAM TCQUE macro	The queue manipulation module processes all requests to add or remove a TCTTE to or from a ZCP activity queue. Additions to the activate queue made by main line modules use compare-and-swap (CS) since an exit routine may also be adding to the queue asynchronously.	None	DFHTCTFX *DFHTCTTE	
DFHZRAC	DFHZRAC1	DFHZATT DFHZDSP	The receive any completion module processes the completion of receive any, sets up the TIOA to be passed to attach, and reissues the RECEIVE ANY macro.	DFHSCP DFHZGET DFHZRST DFHZSDR DFHZUCT	*DFHCSADS *DFHTCADS *DFHTCPRA DFHTCTFX *DFHTCTTE	
DFHZRAQ	DFHZRAQ1	DFHZARQ	The read ahead queuing module is used to save the inbound data stream on temporary storage when an interlock is caused by both the host and the terminal wishing to send data at the same time.	DFHKCP DFHSCP DFHTSP	*DFHCSADS *DFHTCADS *DFHTCTTE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZRAR	DFHZRAR1	DFHZARQ	The read ahead retrieval module is called to retrieve data previously saved on temporary storage by DFHZRAQ.	DFHKCP DFHTSP	*DFHCSADS *DFHTCADS *DFHTCTTE DFHTIOA	
DFHZRLG	DFHZRLNA	DFHZACT	The response logger program logs responses received for protected data sent to an APB. DFHZRLG processes TCTTEs on the log queue when attached by DFHZACT.	DFHJCP	*DFHCSADS *DFHJCADS *DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZRLX	DFHZRLX1	VTAM	DFHZRLX is a VTAM exit routine that handles the completion of LU6.2 RECEIVE requests.		DFHTCTTE	
DFHZRRX	DFHZRRX1	VTAM	The release request exit module receives control from VTAM when another application program has requested connection to a terminal currently connected to CICS. If the terminal is not busy, a CLSDST request is queued to the activate chain. Otherwise the release request indicator is set and the request is processed later by module DFHZDET.	None	DFHTCTFX *DFHTCTTE	
DFHZRSP	DFHZRSNA		The resync send program performs 3614-dependent actions and is also used to retransmit committed output messages. The message is retrieved from temporary storage if necessary.	DFHPCP DFHSCP DFHTCP DFHTSP	*DFHJCRDS *DFHTCADS *DFHTCTTE	
DFHZRST	DFHZRST1	DFHZACT DFHZDET DFHZEMW DFHZNCE	The RESETSR module changes the mode of a session with a terminal and cancels unsatisfied RECEIVE requests. The mode that is set can be Continue Any (CA) or Continue Specific (CS) and RTYPE=DFSYN, DFASY, or RESP.	VTAM	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZRSY	DFHZRSY1	DFHZACT	The resynchronize module resynchronizes CICS and other nodes of the network. DFHZRSY ensures that inbound and outbound sequence numbers are valid.	DFHZRST DFHZSES	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZRVL	DFHZRVL1	DFHZARL	DFHZRVL processes RECEIVE commands for LU6.2 sessions.		DFHTCTTE	
DFHZRVS	DFHZRVS1	DFHZACT	The receive specific module initiates a DFSYN receive specific to obtain the next logical record from a node when a user application requests a DFHTC read. DFHZLRP is called if deblocking is required.	DFHZGET DFHZHPSR DFHZLRP DFHZSDR	*DFHTCADS DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZRVX	DFHZRVX1	VTAM	The receive specific exit module receives control from VTAM when a receive specific is completed. If the data received is too long for the TIOA provided, the overlength data flag is turned on in the TCTTE and the TCTTE is put back on the activate chain. Otherwise, the response is checked and marked in the TCTTE. The data length is set in the TIOA and the FMH is removed.	None	*DFHCSADS DFHTCTFX *DFHTCTTE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZSAX	DFHZSAX1	VTAM	The send DFASY exit module receives control from VTAM when an asynchronous command has completed. It places the TCTTE on the NACP queue if recovery is needed.	None	*DFHTCTTE	
DFHZSCX	DFHZSCX1	VTAM	The SCIP exit module is entered whenever the following asynchronous commands are received: • BIND (as secondary) • UNBIND (as secondary) • STSN (as secondary) • clear (as secondary) • SDT (as secondary) • request recovery (as primary). The module correlates BINDs to a TCTTE and schedules DFHZOPN to complete the BIND process. For the other commands it takes appropriate action and then schedules DFHZNAC using the NACP queue.	None	*DFHTCTTE	
DFHZSDA	DFHZSDA1	DFHZACT	The send data flow asynchronous module handles asynchronous command requests. It ensures that an RPL is allocated, primes the RPL for the requested command and issues the VTAM asynchronous send macro.	None	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZSDL	DFHZSDL1	DFHZARL	DFHZSDL processes SEND commands for LU6.2 sessions.		DFHTCTTE	
DFHZSDR	DFHZSDR1	DFHZACT	The send response module sends responses to nodes when a synchronization request for a terminal is made and a response is outstanding from a previous operation. If errors occur during task initiation, this module is responsible for the negative response.	None	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZSDS	DFHZSDS1	DFHZACT	The send data synchronous module sets up and issues the appropriate VTAM send macro for requests of send data or an SNA synchronous command.	DFHZHPSR DFHZSDR	*DFHTCADS DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZSDX	DFHZSDX1	VTAM	The send data synchronous exit module receives control from VTAM when a send is complete. It checks the RPL for successful completion of the message sent and takes appropriate action.	None	*DFHTCTTE	
DFHZSES	DFHZSES1	DFHZACT DFHZRSY	The session control module is entered whenever a session control command is requested by CICS. It sets up and issues the VTAM SESSIONC command.	None	*DFHTCADS DFHTCTFX *DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZSEX	DFHZSEX1	VTAM	The SESSIONC exit module receives control from VTAM when a SESSIONC command has completed. If the command was successful it turns off the corresponding flags and enqueues the TCTTE to the activate chain. If not successful completion, the TCTTE is placed on the NACP queue for recovery processing.	None	*DFHTCTTE	
DFHZSHU	DFHZSHU1	DFHZDSP	The close VTAM ACB module is invoked whenever CICS and VTAM are being uncoupled. This may be as a result of DFHZTPX being driven as the result of a VTAM halt command or the issue of master terminal command VTAM,CLOSE [,IMMED]. The status of all sessions is checked, and when all are inactive, the ACB is closed. LU6.2 sessions can be terminated by mode name.	DFHKCP DFHSCP DFHZSDS DFHZSIM	*DFHCSADS	
DFHZSIM	DFHZSIM1	DFHZACT	The simulate logon module is entered to issue a VTAM SIMLOGON or REQSESS (if secondary) request to place a node in session without the operator having to logon. LU6.2 can be selected by mode name.	None	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZSIX	DFHZSIX1	VTAM	Whenever a SIMLOGON or REQSESS command has been completed, this exit routine is scheduled by VTAM. On successful completion it turns off the SIMLOGON requested flag and enqueues the TCTTE or TCTME to the activate chain, or if NACP is required, for NACP processing.	None	*DFHTCTTE	
DFHZSKR	DFHZSKR1		The send command response module sends responses to VTAM commands including response to BIND, STSN, and SDT. A positive or negative response can be sent. The module is for secondary LU support only.	None	*DFHTCADS DFHTCTFX *DFHTCTTE	
DFHZSLS	DFHZSLS1	DFHZDSP DFHZOPA	The SETLOGON start module issues SETLOGON to cause VTAM to accept automatic logon requests and issues the initial RECEIVE-ANYS for RPLs in the receive-any pool. DFHZSLS also examines the SIT to determine whether autodefine is used. If it is, the appropriate SIT parameters are copied to the TCT prefix.	DFHZGET VTAM	*DFHTCADS *DFHTCPRA DFHTCTFX IFGRPL	
DFHZSLX	DFHZSLX1	VTAM	DFHZSLX is a VTAM exit routine that handles the completion of LU6.2 SEND requests.		DFHTCTTE	
DFHZSSX	DFHZSSX1	VTAM	The send data flow synchronous exit module receives control when the send of an asynchronous command has been completed.	None	*DFHTCTTE	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZSTU	DFHZSTU1	TC CTYPE= STATUS requests EM* MTP* DFHZLUS	DFHZSTU changes the status of TCTTES and TCTSES. It deals with: • Inservice • Outservice • Intlog | No intlog • Page | Autopage • ATI | NATI.	DFHALP DFHKCP	None	
DFHZSUP	DFHZSUP1	DFHKCP	The startup task module is the entry point for all terminal-related tasks. DFHZSUP performs the following functions: • Sets up the TCTTE status. • Performs security checking. • Performs logging of the TCTTE status and input TIOA. • Performs PCT option checking. • Passes control to user application or to ACP.	DFHJCP DFHKCP DFHSCP DFHSPP DFHXSP DFHZARL	*DFHDWEDS *DFHJCADS *DFHSEC *DFHTCADS *DFHTCTTE DFHTIOA	User pgm DFHACP (through DFHPCP)
DFHZSYN	DFHZSYSN	DFHSPP	DFHZSYN handles CTYPE=SYN and recover requests. For protected message support, DFHSPP issues CTYPE=SYN to flush protected messages. For recover requests, DFHZSYN ensures that no further I/O is issued to that session, and that UNBIND flows.	None	None	
DFHZSYX	DFHZSYX1	VTAM	The SYNAD exit module receives control from VTAM when a catastrophic error is encountered. DFHZSYX determines the type of error and appropriate action to be taken and schedules NACP using the NACP queue to complete the recovery processing.	None	DFHTCTFX *DFHTCTTE	
DFHZTAX	DFHZTAX1	VTAM	The turnaround exit module is called by VTAM on completion of the SEND operation initiated by DFHZRVS in order to perform a turnaround in flip-flop protocol.	None	*DFHTCTTE	
DFHZTPX	DFHZTPX1	VTAM	The TPEND exit module receives control when VTAM is terminating. It schedules a CLSDST for each active session if quick shutdown is required, and sets bits in the TCT prefix so that DFHZSHU will be invoked.	None	DFHTCTFX *DFHTCTZE	
DFHZTRA	DFHZTRA1	DFHZACT DFHZDET DFHZSDR DFHZSDS	DFHZTRA creates appropriate trace entries for the invoking ZCP module, and then creates appropriate VIO trace entries.	DFHTRP	None	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZTSP	DFHZTSP1	DFHCRP DFHISP DFHRTE DFHSPP DFHTPS DFHZARQ DFHZCQ DFHZSUP	The transaction routing program acquires a TCTTE for a link to a remote CICS address space, and transfers request data to that space. DFHZTSP also receives requests from the remote address space.	DFHALP DFHKCP DFHMCP DFHPCP DFHSCP DFHSPP DFHXTP DFHZARQ DFHZISP DFHZLOC	*DFHAIDDS *DFHCSADS DFHEITSP DFHFMHDS *DFHISCRQ DFHLFMDS *DFHPCTDS *DFHTCADS DFHTCTFX DFHTCTSK *DFHTCTTE DFHTIOA DFHXTSDS	
DFHZUCT	DFHZUCT1	DFHZACT DFHZRAC	The uppercase translate module converts a VTAM 3270 data stream into uppercase.	None	DFHTCTFX *DFHTCTTE DFHTIOA	
DFHZUIX	DFHZUIX1	DFHZACT DFHZATT DFHZSDS	The user input exit module uses BALR 14,14 to link to a user-supplied exit routine. It can either be a input, output or an attach routine.	None	DFHTCTFX	
DFHZUSR	DFHZUSR1	DFHZARL DFHZSUP	Maintains the conversation state for LU6.2.	None	DFHTCTTE	
DFHZXCU	DFHZXCU		The VTAM XRF catch-up program, used to send messages that allows a new alternate system to catch up with the current state of the active system for: TCT contents Bound/unbound state of sessions. The program is invoked when a new alternate system signs on.	DFHCCP DFHWMS DFHZXST	DFHTCTTE	
DFHZXQO	DFHZXQO	DFHTCRP DFHZXST	XRF ZCP tracking queue organiser, which allows pending XRF tracking activity to be stored in a way that honors interdependencies, while allowing such requests to be met as soon as all their prerequisites are fulfilled. This component consists of a data structure, and accessing program, using the CICS catalog key structure to identify all the actions for a single resource, and the dependencies between them. Actions are put into the structure on receipt in DFHTCRP, and removed by DFHTCRP, and at the end of DFHZNAC processing for standby BIND and CLSDST completion. The structure is freed at the end of DFHTCRP tracking.	DFHTCRPC DFHTCRPS	DFHZXQOS	
DFHZXRC	DFHZXRC1	DFHZACT	Analyses the data received in response to the SessionC Control=Switch command. Determines the state of the session at the point when it was switched, and initiates the necessary action to clean up, and recover, the session.		DFHTCTFX DFHTCTTE DFHTIOA	

MODULE	ENTRY POINTS	CALLED BY	DESCRIPTION	CALLS	CONTROL BLOCKS REFERENCED	RETURNS TO
DFHZXRE0	DFHZXRE0	System	Runs the transaction CXRE to perform autoconnect and XRF reconnect processing. It also starts the acquire process for terminals with flag TCTEXRE set.			
DFHZXST	DFHZXST	DFHETC DFHSIJ1 DFHTCRP DFHTCRPS DFHZNAC DFHZOPA DFHZXCU	XRF ZCP session-state tracking. Called by DFHZNAC for BIND/UNBIND completion in the active system, and for standby-BIND and UNBIND in the alternate system. Called by DFHETC to track logon-data-freed in the active system. Called by DFHTCRPS to handle a tracking message. Called by DFHTCRP to terminate session tracking. Called by DFHZXCU for BIND/UNBIND catch-up in the active system. Called by DFHSIJ1 and DFHZOPA to issue a SETLOGON START command.	DFHWMS DFHZCP DFHZXSTS	DFHTCTTE	

Chapter 3.3. Control Block Copybook and Macro Names

In general, CICS DSECTs are not coded directly into the source code of CICS modules. Instead, assembler-language copy instructions and CICS macro instructions are used to cause the required DSECTs to be included when the module is being assembled.

The information given in the following table (Figure 203 on page 474) enables the names of the included DSECTs to be determined from a knowledge of the copybook and macro instruction names, and is intended to be used in conjunction with the **Control Blocks Referenced** column in "Chapter 3.1. Module Organization" on page 347.

The table is used in the following way:

1. Find the copybook or macro name in the first column. The names are in alphabetic order.
2. Read the DSECT and/or second-level copybook and macro names in the second column. Second-level names are indicated by an asterisk (*).
3. Reenter the first column with each second-level name, and repeat the process until no further second-level names are found in the second column.

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHAFCD	DFHAFCB
DFHAIDDS	DFHAIDDS DFHAL*
DFHAL	DFHAIDDS DFHEICDS DFHICEDS
DFHALTDS	DFHALTDS
DFHCDBLK	
DFHCNSDS	DFHCNSDS
DFHCRBDS	DFHCRBDS DFHISCRQ*
DFHCSAD	CSAOPFL CSASCBDS DFHCSADS DFHSSADS
DFHCSADS	CSAOPFL CSASCBDS DFHCSAD* DFHCSADS DFHSSADS
DFHDBLDS	DFHDBLDS
DFHDBO	DFHDBODS
DFHDBODS	DFHDBODS
DFHDBRDS	DFHDBRDS
DFHDCADS	DFHDCADS
DFHDCO	DFHDCODS
DFHDCRDS	DFHDCRDS
DFHDCTD	DFHDCTDS DCTSDSCI
DFHDCTDS	DFHDCTD*
DFHDIBDS	DFHDIBDS
DFHDLIAL	DFHDLPDS
DFHDLP	DFHDLPDS DFHISBDS DFHISCRQ* DFHRPDDS DFHRSADS DFHRSBDS DFHXFSTG* DLDO DLMT1 DLMT2 UD
DFHDRCA	DFHDRWA DFHREGS* DFHXFSTG* SDWA
DFHDWCMN	DFHEICDS
DFHDWEDS	DFHAL* DFHDWCMN* DFHDWEDS
DFHDWRMI	DFHUEXIT
DFHEIMDS	DFHISCRQ*
DFHEIPPL	DFHCSAD* DFHEIS* DFHTCA*
DFHEIS	DFHEIBLK
DFHFBO	DFHFBODS

Figure 203 (Part 1 of 7). CICS Control Block and DSECT Names

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names				
DFHFBWA	DFHFBWA				
DFHFCENT	DFHFCENT				
DFHFCTDS	DFHBNDDS	DFHFCTDS	DFHFPFDS		
DFHFCTSR	DFHFCTSR				
DFHFIOA	DFHFIOA				
DFHFMHDS	DFHFMHDS				
DFHFWADS	DFHFWADS				
DFHHTADS	DFHHTADS				
DFHICEDS	DFHAL*	DFHICEDS			
DFHIMSDS	DFHIMSDS				
DFHIRPD	AXAE LCB SCTE	CCB LCBE SUDB	CSB LQCELL UCA	LACB LQCL UCAE	LACBE LXAE
DFHIRRDS	DFHIRRDS				
DFHIRSDS	IRSVCADS SCCB	IRSVCFDS SLCB	LCL	SCACB	SCACBE
DFHISCRQ	DFHISCRQ				
DFHJCA	DFHJCADS	DFHJCN			
DFHJCADS	DFHJCA*				
DFHJCICA	DFHJCICA				
DFHJCN	None				
DFHJCOCL	DFHJCOCL				
DFHJCR	DFHJCRDS				
DFHJCRDS	DFHJCR*	DFHJCRDS			
DFHJCTDS	DFHJCT				
DFHJCTTE	DFHJCEDD	DFHJCEXD	DFHJCTTE	JCTLRN	JCTSQE
DFHKCTWA	DFHKCTWA				
DFHKPBUF	KPBUF				
DFHKPPDS	DFHKPPDS				
DFHKPTDS	DFHKPTDS				
DFHKPTE	DFHKPTE				
DFHLFM	DFHLFS	DFHLLADS	DFHLPLST	DFHREGS*	

Figure 203 (Part 2 of 7). CICS Control Block and DSECT Names

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHLLADS	DFHLLADS
DFHLLDC	DFHLLDC
DFHLUCDS	DFHLUCDS
DFHLUMDS	DFHLUMDS
DFHMAPDS	DFHMAPDS
DFHMBCDS	DFHMBCA DFHMBCB DFHMQCB
DFHMBO	DFHMBODS
DFHMCAD	DFHMCADS
DFHMCBDS	DFHMCB
DFHMCIN	DFHMCIN
DFHMCPE	DFHMCPE
DFHMCRDS	DFHMCRDS ISTDNIB
DFHMGM	ETMGDSCT ETMGTEXT MGINSERT MGMAMAP MGMMDEST
DFHMRCDS	DFHMRCA DFHMRCB
DFHOCLDS	DFHOCLDS
DFHOCODS	DFHOCODS
DFHOSPWA	DFHOSPWA
DFHPAMDS	DFHPAM
DFHPCTDS	DFHPCTDS DFHPCTDX DFHPCTIL DFHPCTPF DFHRMTDX
DFHPGADS	DFHPGADS
DFHPGLST	DFHPGLST
DFHPLTDS	DFHPLTDS
DFHPPTDS	DFHPPTDS DFHPPTDX
DFHPRADS	DFHPRADS
DFHPRINT	DFHTCTLE DFHTCTSK
DFHQCADS	DFHQCADS DFHQEADS
DFHREGS	DFHDWEDS*
DFHSAADS	DFHSAADS

Figure 203 (Part 3 of 7). CICS Control Block and DSECT Names

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHSC	DFHSAADS DFHPAM DFHSCCOS DFHSCIDS DFHSCXDS DFHPRADS SPHEADER DFHFAQE
DFHSCCOS	DFHSECDS
DFHSEC	DFHSECDS
DFHSICOM	DFHAIDDS* DFHALTDS DFHCSAD* DFHDCTDS* DFHFCTDS* DFHICEDS* DFHJCOCL DFHJCTDS* DFHJCTTE* DFHKPBUF* DFHKPPDS DFHPAMDS* DFHPCTDS* DFHPLTDS DFHPPTDS* DFHSIPD* DFHSIT* DFHTCA* DFHTCTFX DFHTCTLE DFHTCTZE* DFHTSCTL DFHTSMDS* DFHZEPD

Figure 203 (Part 4 of 7). CICS Control Block and DSECT Names

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHSIPD	DFHSIPDS
DFHSIPDS	DFHSIPD* DFHSIPDS
DFHSIT	DFHLISTA DFHSITDS DFHSITDS
DFHSLDC	DFHSLDC
DFHSNT	DFHSNNT
DFHSRTDS	DFHSRTDS
DFHSTR	DFHSTRDS STRTE
DFHTACB	DFHABND
DFHTACLE	DFHTCTLE
DFHTBOD	DFHTBODS
DFHTBODS	DFHTBOD* DFHTBODS
DFHTCA	DFHAL* DFHBMS DFHDC DFHDI DFHFC DFHIC DFHKC DFHKP DFHOC DFHPC DFHPS DFHSC* DFHSP DFHTC DFHTCADY DFHTD DFHTR DFHTS
DFHTCADS	DFHTCA*
DFHTCPCM	ZATHACDS ZATHAIDS ZATHMJTB ZATHPGDS
DFHTCPRA	DFHTCPRA DFHTCTLE IFGRPL
DFHTCPSV	DFHAIDDS* DFHCSAD* DFHDCTDS* DFHDIBDS DFHDWEDS* DFHFMHDS DFHISCRQ* DFHJCA* DFHLFM* DFHMGM* DFHPCTDS* DFHSIT* DFHTACB* DFHTCA* DFHTCPRA* DFHTCTFX DFHTCTZE* DFHTIOA DFHUEXIT* DFHVTWA* DFHXLTDS DFHZEPD

Figure 203 (Part 5 of 7). CICS Control Block and DSECT Names

Chapter 4.2. CICS Source Modules

This chapter contains an alphabetical list of all CICS source modules.

Name	Type	Description	Reference	Library
CALLDLI	Macro	CALL DL/I services	User	MCP
CSAOPFL	DSECT	CSA optional features list	User	
DFHABEND	Macro	Issues an ABEND macro	System	
DFHACEE	CSECT	Security block search		M
DFHACP	CSECT	Abnormal condition program	Figure 34	M
DFHAICB	Macro	Application interface control block	System	M
DFHAID	Symbolic	3270 attention identifiers	User	MCP
DFHAIDDS	DSECT	Automatic initiate descriptor	User	M
DFHAKP	CSECT	Activity keypoint program	Figure 36	S
DFHALP	CSECT	Terminal allocation		M
DFHALT	Macro	Application load table	Table gen	M
DFHALTDS	DSECT	Application load table	DFHSIP	M
DFHAMP	CSECT	Allocation management program		O
DFHAMTP	CSECT	Request processor for DFHAMP	DFHAMP	O
DFHANRAT	Macro	3270 attribute char resolution	DFHMDF	M
DFHANRWC	Macro	3270 control char resolution	DFHMDI	MCP
DFHAUTH	Macro	Verifies environment & activates CICS SVCs	System	M
DFHBFP	CSECT	Built-in functions program		
DFHBFPC	CSECT	Built-in function program		S
DFHBFTCA	Macro	Built-in functions TCA macro	User	MCP
DFHBIF	Macro	Built-in function request	User	MCP
DFHBIFBA	Source	Basic TCA for BIF	DFHBFP	CP
DFHBIFWR	Symbolic	Built-in function definitions	User	CP
DFHBLLDS	DSECT	COBOL base registers	User	C
DFHBMPIC	Macro	BMS picture analysis	DFHMDF	M
DFHBMS	Macro	Basic mapping support request	User	MCP
DFHBMSCA	Symbolic	BMS attribute definitions	User	MCP
DFHBMSKS	Macro	Print line of asterisks	User	M
DFHBMUTM	Macro	Trace DFHSG BMS options	System gen	M
DFHBSC	Macro	Generates binary search code	Table gen	M
DFHBSHDR	Macro	Call builder macro	System	M
DFHBSIB3	CSECT	BMS 3270 builder		O
DFHBSIZ3	CSECT	Add 3270 support		O
DFHBSMIR	CSECT	Build terminal session		O
DFHBSMPP	CSECT	Build pipeline pool table entry		O
DFHBSMSG	Macro	Builder message macro	System	M
DFHBSM61	CSECT	Generate sessions for modegroup		O
DFHBSM62	CSECT	Build a modegroup		O
DFHBSPTE	Macro	Builder pattern table entry macro		M
DFHBSS	CSECT	Build a connection		O
DFHBSSA	CSECT	Pass a new TCT entry to DFHKCP		O
DFHBSSF	CSECT	Pass a new TCT entry to DFHSTPD		O
DFHBSSS	CSECT	Pass a new TCT entry to DFHXSP		O
DFHBSSZ	CSECT	Pass a new TCT entry to DFHZCP		O
DFHBSSZB	CSECT	Add a new batch IRC connection		O
DFHBSSZG	CSECT	Add an APPC single-session		O
DFHBSSZI	CSECT	Add an indirect terminal system		O
DFHBSSZL	CSECT	Add a local terminal system		O
DFHBSSZP	CSECT	Add an APPC parallel-session		O
DFHBSSZR	CSECT	Add an MRO system		O
DFHBSSZS	CSECT	Add an APPC		O
DFHBSSZ6	CSECT	Add an LU6.1 connection		O

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHTCTCE	DFHTCTCE
DFHTCTFX	DFHTCTFX
DFHTCTLE	DFHTCTLE
DFHTCTTE	DFHTCTZE*
DFHTCTWA	DFHTCTTE* TCTENIB TCTEQNAM TCTTETTE
DFHTCTZE	DFHTCTTE TCTENIB TCTEQNAM TCTTETTE
DFHTDCI	DFHTDCI
DFHTDIA	DFHTDIA
DFHTDOA	DFHTDOA
DFHTDSDS	DFHTDST
DFHTIEDS	DFHDWCMN DFHDWRMI
DFHTIOA	DFHTIOA

Figure 203 (Part 6 of 7). CICS Control Block and DSECT Names

Copybook or Macro Name	DSECT or Second-Level Copybook or Macro Names
DFHTRACE	ZTRENTRY ZTRHEADR
DFHTRADS	DFHTRADS
DFHTSBDS	DFHTSACA DFHTSBCA DFHTSVCA DFHTSREB DFHTSQE DFHTSRE
DFHTSCI	DFHTSCI
DFHTSCTL	DFHTSCTL
DFHTSHD	DFHTSIOA*
DFHTSIOA	DFHTSIOA
DFHTSMDS	DFHTSGID DFHTSMAP DFHTSUT DFHTSUTE
DFHTSTDS	DFHTSTDS
DFHTTPDS	DFHTTPCM DFHTTPRE
DFHTUL	DFHTULDS
DFHUEXIT	DFHEPB DFHEPL DFHUETE DFHUETH
DFHURLDS	DFHURLDS
DFHVC	DFHVCFLG DFHVCRQ
DFHVCFLG	DFHVCFLG
DFHVOLD	DFHSERIS DFHVOLAN DFHVOLDS* VOLINDEX
DFHVOLDS	DFHVOLDS
DFHVSWA	DFHVSWA
DFHVTWA	DFHTCADS
DFHXFIOA	DFHXFIOA
DFHXFMOD	DFHCSAD* DFHEIS* DFHFMHDS DFHLFM* DFHSEC* DFHTCA* DFHTCTZE* DFHTIOA
DFHXFSTG	DFHXFRDS
DFHXLTDS	DFHXLTDS
DFHXTSTG	DFHXTSDS
DFHZAIT	DFHTCPCM*
DFHZEPD	DFHZEPD

Figure 203 (Part 7 of 7). CICS Control Block and DSECT Names

Part 4. Directory

This part provides a directory of all CICS source modules.

The part contains the following chapters:

- Introduction
- CICS Source Modules.

Chapter 4.1. Introduction

This chapter contains a description of the various types of modules distributed with CICS. "Chapter 4.2. CICS Source Modules" on page 485 contains an alphabetical list of all source modules.

The types of modules are:

Macro A macro definition.

DSECT A dummy section defining a CICS data area.

Symbolic Definition of a CICS data area (with no DSECT statement), or a group of EQU statements which symbolically define values used throughout a program.

CSECT Source code for a control section or the first part of a control section (other source modules may be copied by the CSECT).

Exit An exit from and a return to a CICS module at a stated functional point. The user can insert code at these points to enhance the program.

Source Source code which is not a CSECT.

Sample Sample programs and tables.

Other Job control language statements, cataloged procedures or object code placed in the source library for convenience in distribution. See the *CICS/MVS Operations Guide* for the handling of these modules.

The significance of the name in the reference column differs with the type of module as shown in the table on page 484.

Type	Reference
Macro	
For CICS system use only	"System"
For use by application programmer	"User"
System generation	"System gen"
Table generation	"Table gen"
Inner macro instruction	Name of outer macro instruction
DSECT or symbolic definition	
For CICS system use only	"System"
For use by application programmer	"User"
Used primarily by one program	Program name
CSECT	
First of program	Figure number
Not first of program	Program name
Sample	"APRM(C)" "CG" "ICG" "OPG"
Source Module	Member name

Notes:

1. *"APRM(C)" is the CICS/MVS Application Programmer's Reference.*
2. *"CG" is the CICS/MVS Customization Guide.*
3. *"ICG" is the CICS/MVS Intercommunication Guide.*
4. *"OPG" is the CICS/MVS Operations Guide.*

CICS Modules

The modules are cataloged as members of a library during system preparation. See the *CICS/MVS Installation Guide*. Some modules have COBOL and/or PL/I equivalents of assembler code by the same name; these modules are shown as cataloged in more than one library. The meanings of the letters in the library column are:

C	Cataloged in CICS.COBLIB
I	Generated by the installation process - DFHINST
M	Cataloged in CICS.MACLIB
O	Optional source
P	Cataloged in CICS.PLILIB
S	Cataloged in CICS.SOURCE
X	Cataloged in CICS.SAMPLIB
Y	Cataloged in CICS.PROCLIB.

Microfiche

The microfiche, which is available with CICS, contains assembled listings of programs and source listings of macro instructions, DSECTs, and symbolic definitions.

Name	Type	Description	Reference	Library
DFHBST	CSECT	Common TCTTE builder		
DFHBSTB	CSECT	Add a resource for BMS		O
DFHBSTBL	CSECT	Add logical device support		O
DFHBSTB3	CSECT	Add partition support		O
DFHBSTC	CSECT	Add install-time options support		O
DFHBSTD	CSECT	Add DFHDIP support		O
DFHBSTE	CSECT	Add DFHEDF support		O
DFHBSTI	CSECT	Add DFHICP support		O
DFHBSTL	CSECT	Add logical device support		S
DFHBSTM	CSECT	Add DFHMGP support		O
DFHBSTP3	CSECT	Add 3270-copy support		O
DFHBSTS	CSECT	Add DFHSNT support		O
DFHBSTT	CSECT	Add DFHKCP support		O
DFHBSTZ	CSECT	Build terminal or session resource		O
DFHBSTZA	CSECT	Add DFHZCP support		O
DFHBSTZB	CSECT	Add or delete bind-image		O
DFHBSTZC	CSECT	Add single-session to APPC		O
DFHBSTZH	CSECT	Add IRC batch session		O
DFHBSTZO	CSECT	Add an OS/VS console		O
DFHBSTZR	CSECT	Add IRC session		O
DFHBSTZS	CSECT	Add an APPC session		O
DFHBSTZV	CSECT	Add VTAM and IRC information		O
DFHBSTZZ	CSECT	Add non-APPC session		O
DFHBSTZ1	CSECT	Add remote terminal support		O
DFHBSTZ3	CSECT	Add 3270 support		O
DFHBSZGB	DSECT	DSECTs for terminal builder program		S
DFHBSZZ	CSECT	Add terminal or session		O
DFHBSZZS	CSECT	Add session to LU6.2 support		O
DFHBSZZV	CSECT	Add VTAM terminal or session		O
DFHBT	Macro	Parameter sublist translation	System	M
DFHBXDL	Macro	PL/I optimizer library macro	System	M
DFHBXER	Macro	PL/I optimizer library macro	System	M
DFHBXGC	Macro	PL/I optimizer library macro	System	M
DFHBXGV	Macro	PL/I optimizer library macro	System	M
DFHBXIN	Macro	PL/I optimizer library macro	System	M
DFHBXLB	Macro	PL/I optimizer library macro	System	M
DFHBXRT	Macro	PL/I optimizer library macro	System	M
DFHBXTA	Macro	PL/I optimizer library macro	System	M
DFHBXTS	Macro	PL/I optimizer library macro	System	M
DFHBXTV	Macro	PL/I optimizer library macro	System	M
DFHCAA70	CSECT	7770 appendage	DFHTCP	S
DFHCAP	CSECT	RDO command analysis utility		S
DFHCCMF	CSECT	Monitoring ATI program		
DFHCCP	CSECT	Catalog control program		O
DFHCCWDS	DSECT	Channel command word definition		M
DFHCDBLK	DSECT	CONVDATA area		S
DFHCICS	CSECT	CICS copyright information		S
DFHCLT	Macro	Command list table	Table gen	M
DFHCMON	CSECT	Monitoring online program		
DFHCMP	CSECT	CICS monitoring program		
DFHCNSDS	DSECT	Change number of sessions control block		S
DFHCNSL	CSECT	Offline utility console handler	DFHDUP DFHSTUP DFHJCJFP	
DFHCOMP	Macro	Generate compare equate values	DFHHASH	M

Name	Type	Description	Reference	Library
DFHCOVER	Macro	Cover page generator	User	MCP
DFHCPY	CSECT	3270 hard copy support	DFHZCP	S
DFHCRBDS	DSECT	CICS region control block	System	M
DFHCRC	CSECT	Interregion abnormal exit module		
DFHCRNP	CSECT	Interregion connection manager		S
DFHCRP	CSECT	Transaction routing relay program	Figure 193	S
DFHCRQ	CSECT	ATI purge program		S
DFHCRR	CSECT	Interregion session recovery program		
DFHCRS	CSECT	Remote scheduler program		S
DFHCRSP	CSECT	CICS IRC startup module		S
DFHCSA	CSECT	Common system area	System	S
DFHCSADS	DSECT	Common system area definition	User	MCP
DFHCSDUP	CSECT	CSD utility program		S
DFHCSSC	CSECT	Security time-out transaction		0
DFHCSVC	Source	CICS SVC		S
DFHCT	Macro	Code translation macro	System	M
DFHCUCA	CSECT	RDO command analyzer		S
DFHCUCB	CSECT	RDO command builder		0
DFHCUCD	CSECT	RDO command default values		S
DFHCUCV	CSECT	RDO command validation		0
DFHCUMIG	CSECT	RDO migration		0
DFHCU210	Source	Resource definition list for CSD		0
DFHCWTO	CSECT	Write to console operator program	System	
DFHDATE	Macro	Date print macro	System	M
DFHDBLDS	DSECT	Dynamic log	User	M
DFHDBO	Macro	DL/I backout table	DFHRUP	M
DFHDBODS	DSECT	DL/I backout table	DFHRUP	S
DFHDBP	CSECT	Dynamic backout program	Figure 33	S
DFHDBRDS	DSECT	Dynamic backout record	User	M
DFHDC	Macro	Dump service request	User	MCP
DFHDCADS	DSECT	Dispatch control area	DFHKCP	M
DFHDCO	Macro	Dump control option list	System	M
DFHDCP	CSECT	Dump control program	Figure 31	S
DFHDCRDS	Symbolic	Dump control record formats	DFHDCP DFHDUP	S
DFHDCT	Macro	Destination control table	Table gen	M
DFHDCTDS	DSECT	Destination control table	DFHTDP	M
DFHDEBDS	DSECT	Data set block definition	System	M
DFHDEB70	CSECT	7770 DEB processor	DFHTCP	S
DFHDESM	Macro	LU6.2 security encryption macro	System	M
DFHDI	Macro	Data interchange request	User	M
DFHDIBDS	Macro	Data interchange	System	M
DFHDIP	CSECT	Data interchange program		S
DFHDITOP	Macro	Data interchange internal macro	System	M
DFHD'BP	CSECT	DL/I backout program		S
DFH'.DBD	Macro	Generate DL/I DMB directory	Table gen	M
DFHDLG	CSECT	Global command task		S
DFHDLI	CSECT	CICS-DL/I interface	Figure 22	S
DFHDLIAI	CSECT	Application interface for DL/I		S
DFHDLIAL	DSECT	DL/I address list	System	S

Name	Type	Description	Reference	Library
DFHDLIDS	DSECT	Reserved for VANDL compatibility	System	M
DFHDLP	Macro	CICS-DL/I interface	System	M
DFHDLQ	CSECT	IMS/VS quasi-application program		S
DFHDLR	CSECT	Service routines for DL/I		S
DFHDLRES	Macro	DL/I remote resource table	System	
DFHDLRP	CSECT	DL/I restart program		O
DFHDLS	CSECT	Status command task		S
DFHDLX	CSECT	Subroutines for DFHDLQ	DFHDLQ	S
DFHDLXA	CSECT	Extension to DFHDLX	DFHDLQ	S
DFHDMP	CSECT	Definition file management program	DFHAMP	S
DFHDRCA	DSECT	Dependent (batch) region control area	System	M
DFHDRP	CSECT	Dependent (batch) region control module		S
DFHDRPA	CSECT	Dependent (batch) region initialization module		S
DFHDRPB	CSECT	Application program control module		S
DFHDRPC	CSECT	Batch region termination module		S
DFHDRPD	CSECT	Batch region cleanup and exits module		S
DFHDRPE	CSECT	DL/I request handling module		S
DFHDRPF	CSECT	SVC initialization module		S
DFHDRPG	CSECT	EXEC DLI stub for shared data bases	DFHEIP	S
DFHDRX	Macro	DL/I resource table	System	M
DFHDSB	CSECT	BMS data stream build	DFHPBP	S
DFHDSCTS	DSECT	System generation print option	System gen	S
DFHDSND	Macro	File control data set name	System	M
DFHDUP	CSECT	Dump utility program		S
DFHDWCMN	DSECT	Common part of DWE	DFHDWEDS	S
DFHDWE	Macro	Deferred work element	Table gen	M
DFHDWEDS	DSECT	Deferred work element	DFHDWEDS	M
DFHDWRMI	DSECT	DWE extension for task-related user exit interface	DFHTIEDS	S
DFHEAI	CSECT	Exec (command-level) interface stub for assembler		S
DFHEAM01	CSECT	Command translator - control module	DFHEAP	S
DFHEAM02	CSECT	Command translator - initialization option	DFHEAP	O
DFHEAM26	CSECT	Command translator - error editor	DFHEAP	O
DFHEAP	CSECT	Command translator for assembler		
DFHEBF	CSECT	Exec interface for built-in functions		S
DFHEBU	CSECT	EXEC FMH construction		S
DFHECB	Macro	CICS posting and testing of operating system ECBs	System	M
DFHECI	CSECT	Exec (command-level) interface stub for COBOL		S

Name	Type	Description	Reference	Library
DFHECM01	CSECT	Command translator - control module		S
DFHECM02	CSECT	Command translator - initialization	DFHECP	O
DFHECM07	CSECT	Command translator - options card	DFHECP	O
DFHECM08	CSECT	Command translator - check options	DFHECP	O
DFHECM09	CSECT	Command translator - print options	DFHECP	S
DFHECM10	CSECT	Command translator - analyze program	DFHECP	O
DFHECM11	CSECT	Command translator - atomize	DFHECP	O
DFHECM12	CSECT	Command translator - match brackets	DFHECP	S
DFHECM14	CSECT	Command translator - read input	DFHECP	O
DFHECM15	CSECT	Command translator - I/O module	DFHECP	S
DFHECM17	CSECT	Command translator - generate output	DFHECP	O
DFHECM26	CSECT	Command translator - error editor	DFHECP	O
DFHECP	CSECT	Command translator for COBOL		
DFHEDC	CSECT	Exec interface for dump control		S
DFHEDFBR	CSECT	Temporary storage browse transaction	DFHEDFD	O
DFHEDFCB	CSECT	Build one page		O
DFHEDFCC	CSECT	Parameter copy program		O
DFHEDFCE	CSECT	Extract from one page		O
DFHEDFCR	CSECT	LD table utilities		O
DFHEDFCS	CSECT	CICS special cases		O
DFHEDFCX	CSECT	Display unformatted arguments		O
DFHEDFD	CSECT	EDF display program	Figure 191	O
DFHEDFDL	CSECT	DL/I special cases		O
DFHEDFDS	DSECT	EDF control information	System	M
DFHEDFM	CSECT	EDF map set	DFHEDFD	O
DFHEDFP	CSECT	EDF control program	Figure 191	O
DFHEDFR	CSECT	EDF response table	Figure 191	O
DFHEDFU	CSECT	Data utilities		O
DFHEDFW	CSECT	Display working storage		O
DFHEDFX	CSECT	EDF task switch program	Figure 191	S
DFHEDI	CSECT	Exec interface for data interchange		S
DFHEDP	CSECT	EXEC DLI command stub	DFHEIP	S
DFHEEI	CSECT	Exec interface for HANDLE, ADDRESS, ASSIGN		S
DFHEEX	CSECT	EXEC FMH extraction		S
DFHEFC	CSECT	Exec interface for file control		S
DFHEGL	CSECT	LU6.2 request program	DFHEIP	S
DFHEIAR	Macro	EIP arguments macro	System	M
DFHEIBLK	DSECT	Exec interface block	User	SCP
DFHEIC	CSECT	Exec interface for interval control		S
DFHEICDS	DSECT	Exec interface COMMAREA	System	M
DFHEIDTI	CSECT	Exec ask-time, format-time program		O

Name	Type	Description	Reference	Library
DFHEIEND	Macro	Exec interface storage end macro	User	M
DFHEIENT	Macro	Exec interface prolog macro	User	M
DFHEIFD	DSECT	EXEC interface file control DSECT	System	M
DFHEIFSP	Macro	Free space	System	M
DFHEIGSP	Macro	Get space	System	M
DFHEIIF	Macro	Exec (command-level) interface IF macro	System	M
DFHEILIA	Catlg Proc	Used by DFHEITAL cataloged procedure	User	S
DFHEILIC	Catlg Proc	Used by DFHEITCL cataloged procedure	User	C
DFHEILIP	Catlg Proc	Used by DFHEITPL cataloged procedure	User	P
DFHEIMSG	Macro	Exec interface message macro	User	M
DFHEIMV	Macro	Exec (command-level) interface MOVE macro	System	M
DFHEIP	CSECT	Exec (command-level) interface program	Figure 11	S
DFHEIPAD	Macro	Exec interface intermodule addressing	DFHEIP	M
DFHEIPDS	Source	Exec interface private DSECTs	DFHEIP	M
DFHEIPEL	Source	Exec interface epilog code	DFHEIP	M
DFHEIPEQ	Symbolic	Exec interface EQU statements	DFHEIP	M
DFHEIPER	Source	Exec interface error handling	DFHEIP	M
DFHEIPLR	Macro	Epilog code	System	M
DFHEIPLS	Macro	Prolog code	System	M
DFHEIPPL	Source	Exec interface prolog code	DFHEIP	M
DFHEIQDS	CSECT	Exec inquire/set for data sets	DFHEIP	O
DFHEIQSA	CSECT	Exec inquire/set for attributes	DFHEIP	O
DFHEIQSC	CSECT	Exec inquire/set for connections	DFHEIP	O
DFHEIQSM	CSECT	Exec inquire/set for modenames	DFHEIP	O
DFHEIQSP	CSECT	Exec inquire/set for programs	DFHEIP	O
DFHEIQST	CSECT	Exec inquire/set for terminals	DFHEIP	O
DFHEIQSX	CSECT	Exec inquire/set for transactions	DFHEIP	O
DFHEIRET	Macro	Exec interface epilog code	User	M
DFHEIS	Macro	Exec interface define exec interface storage	System	M
DFHEISDS	DSECT	Exec interface define exec interface storage	System	M
DFHEISTG	Macro	Exec interface storage start macro	User	M
DFHEITAB	DSECT	Translator table	DFHEIP	O
DFHEITAL	Other	Cataloged procedure to translate, assemble, and link-edit assembler language programs	User	
DFHEITCL	Other	Cataloged procedure to translate, compile, and link-edit COBOL language programs	User	
DFHEITCU	DSECT	RDO offline LD table		O
DFHEITPL	Other	Cataloged procedure to translate, compile, and link-edit PL/I language programs	User	
DFHEITSP	Source	Language definition table		O

Name	Type	Description	Reference	Library
DFHEITUT	Source	Definition of EIP trace entries	System	S
DFHEIVAR	DSECT	COBOL working storage	User	C
DFHEJC	CSECT	Exec interface for journal control		S
DFHEJECT	Macro	Page eject/space option	System	M
DFHEKC	CSECT	Exec interface for task control		S
DFHELR	CSECT	ISC local/remote determination		S
DFHEMA	CSECT	Enhanced master terminal		
DFHEMB	CSECT	Enhanced master terminal		
DFHEMC	CSECT	Enhanced master terminal		
DFHEMD	CSECT	Enhanced master terminal		
DFHEME	CSECT	Enhanced master terminal		
DFHEMF	CSECT	Enhanced master terminal		
DFHEMG	CSECT	Enhanced master terminal		
DFHEMH	CSECT	Enhanced master terminal		
DFHEMI	CSECT	Enhanced master terminal		
DFHEMS	CSECT	Exec interface for BMS		S
DFHEND	Macro	Generates END statement	System	M
DFHEPC	CSECT	Exec interface for program control		S
DFHEPI	CSECT	Exec (command-level) interface stub for PL/I		S
DFHEPM01	CSECT	Command translator - control module		S
DFHEPM02	CSECT	Command translator - initialization	DFHEPP	O
DFHEPM07	CSECT	Command translator - options card	DFHEPP	O
DFHEPM08	CSECT	Command translator - check options	DFHEPP	O
DFHEPM09	CSECT	Command translator - print options	DFHEPP	S
DFHEPM10	CSECT	Command translator - analyze program	DFHEPP	O
DFHEPM11	CSECT	Command translator - atomize	DFHEPP	O
DFHEPM12	CSECT	Command translator - match brackets	DFHEPP	S
DFHEPM14	CSECT	Command translator - read input	DFHEPP	O
DFHEPM17	CSECT	Command translator - generate output	DFHEPP	O
DFHEPM26	CSECT	Command translator - error editor	DFHEPP	O
DFHEPP	CSECT	Command translator for PL/I		S
DFHEPS	CSECT	System spooling interface stub	DFHEIP	O
DFHERM	CSECT	Non-CICS licensed program	DFHEIP	S
DFHESC	CSECT	Exec interface stub for storage control	DFHEIP	S
DFHESP	CSECT	Exec interface stub for sync point control	DFHEIP	S
DFHETC	CSECT	Exec interface stub for terminal control	DFHEIP	S
DFHETD	CSECT	Exec interface stub for transient data	DFHEIP	S
DFHETL	CSECT	LU6.2 EXEC interface stub	DFHEIP	S

Name	Type	Description	Reference	Library
DFHETR	CSECT	Exec interface stub for trace control	DFHEIP	S
DFHETS	CSECT	Exec interface stub for temporary storage	DFHEIP	S
DFHEXI	Source	3270 hard copy support	DFHZCP	S
DFHEXMF4	CSECT	Command translator - FULL tables	DFHEXPA	S
DFHEXMPE	CSECT	Preprocessor error interface	DFHPRPR	O
DFHEXMS4	CSECT	Command translator - subset tables	DFHEXP	S
DFHEXM03	CSECT	Command translator - storage allocation	DFHEXP	S
DFHEXM05	CSECT	Command translator - PARM analysis	DFHEXP	S
DFHEXM13	CSECT	Command translator - note diagnostic	DFHEXP	S
DFHEXM15	CSECT	I/O module		
DFHEXM16	CSECT	Command translator - conversions	DFHEXP	S
DFHEXM18	CSECT	Command translator - insert in I/O buffer	DFHEXP	S
DFHEXM25	CSECT	Command translator - print XREF	DFHEXP	S
DFHEXM26	CSECT	Messages table		
DFHEXM27	CSECT	Command translator - spelling correction	DFHEXP	S
DFHFBO	Macro	File backout table	DFHRUP	M
DFHFBODS	DSECT	File backout table	DFHRUP	S
DFHFC	Macro	File service request	User	MCP
DFHFCBP	CSECT	File backout program		S
DFHFCC	CSECT	File control open, close	DFHFCP	O
DFHFCD	CSECT	File control program	DFHFCP	O
DFHFCENT	DSECT	File control operation entry	System	M
DFHFCEXT	Exit	File control exit program	DFHFCP	S
DFHFCISA	DSECT	CICS modified ISMOD save area	DFHFCP	S
DFHFCJ	CSECT	File control journaling	DFHFCP	O
DFHFCL	CSECT	VSAM LSR pool builder	DFHFCP	O
DFHFCN	CSECT	File control open/close program	DFHFCP	O
DFHFCNC	Source	File close program	DFHFCN	O
DFHFCNO	Source	File open program	DFHFCN	O
DFHFCP	CSECT	File control program	Figure 21	O
DFHFCRP	CSECT	File control restart program	DFHFCP	O
DFHFCS	CSECT	File control state, attribute change	DFHFCP	O
DFHFCT	Macro	File control table	Table gen	M
DFHFCTDS	DSECT	FCT data set control	DFHFCP	S
DFHFCTSP	Macro	Shared resource control block	System	M
DFHFCTSR	DSECT	VSAM shared resources	DFHFCP	S
DFHFCU	CSECT	File open utility program	DFHFCP	O
DFHFCV	CSECT	File control VSAM requests	DFHFCP	O
DFHFCX	DSECT	File control UPAD exit	DFHFCP	O
DFHFCWSS	DSECT	File control work areas	DFHFCP	S
DFHFD	Macro	Formatted dump request system	System	M
DFHFDB	CSECT	Formatted dump program table interpretation	DFHFDP	S
DFHFDC	CSECT	Formatted dump program tables	DFHFDP	S
DFHFDD	CSECT	Formatted dump program	DFHFDP	

Name	Type	Description	Reference	Library
DFHFDP	CSECT	Formatted dump program	User	S
DFHFDPDS	DSECT	Formatted dump program	DFHFDP	S
DFHFEP	CSECT	Field engineering program		S
DFHFIOA	DSECT	File input/output area	User	MCP
DFHFMH	Macro	Function management header	DFHZCP	M
DFHFMHDS	DSECT	Function management header	DFHZCP	M
DFHFMIDS	Symbolic	Function and module identifiers	User	M
DFHFTAP	CSECT	Format tape program		S
DFHFWADS	DSECT	File work area	User	MCP
DFHGEN	Macro	System generation	System	
DFHGMM	CSECT	VTAM LU startup message		S
DFHHASH	Macro	Locate TCTTE entries	DFHTCP	M
DFHHPSVC	CSECT	HPO type 6 SVC		S
DFHHTADS	DSECT	HPO transaction area	System	M
DFHIC	Macro	Time service request	User	MCP
DFHICEDS	DSECT	Interval control element	DFHICP	M
DFHICP	CSECT	Interval control program	Figure 14	S
DFHIIP	CSECT	Non-3270 input mapping program		S
DFHIMSDS	DSECT	ISC message inserts	System	M
DFHIOBDS	DSECT	Input/output block	System	M
DFHIR	Macro	Interregion macro	System	M
DFHIRP	CSECT	Interregion communication program		
DFHIRSDS	DSECT	Interregion subsystem control block	System	M
DFHIS	Macro	ISC request macro	System	M
DFHISCRQ	Macro	ISC request parameter list	System	M
DFHISP	CSECT	ISC request shipping		S
DFHJC	Macro	Journal service request	User	MCP
DFHJCA	Macro	Journal control area definition	DFHJCP	M
DFHJCADS	DSECT	Journal control area	User	MCP
DFHJCBSP	CSECT	Journal tasks bootstrap program	DFHJCP	S
DFHJCC	CSECT	Journal control close	DFHJCPA	S
DFHJCEOV	CSECT	Journal control EOV	DFHJCP	S
DFHJCI	CSECT	Journal control input	DFHJCP	S
DFHJCICA	DSECT	Journal control DECB	DFHJCP	M
DFHJCIOE	CSECT	Journal control I/O error program	DFHJCP	S
DFHJCJFP	CSECT	Journal control format program	DFHJCP	S
DFHJCKOJ	CSECT	Journal control kickoff program	DFHJCP	S
DFHJCO	CSECT	Journal control open	DFHJCP	S
DFHJCOCL	DSECT	Journal control open/close list	DFHJCP	M
DFHJCOCP	CSECT	Journal control open/close program	DFHJCP	S
DFHJCP	CSECT	Journal control program	Figure 26	S
DFHJCR	Macro	Journal control record	DFHJCP	M
DFHJCRDS	DSECT	Journal control record	User	M
DFHJCRP	CSECT	Journal control recovery program	System	O
DFHJCSDJ	CSECT	Journal control shutdown	DFHJCP	S
DFHJCT	Macro	Journal control table	Table gen	M
DFHJCTDS	DSECT	Journal control table	DFHJCP	M
DFHJCTTE	DSECT	Journal control table - entry	DFHJCP	M
DFHJUP	CSECT	Journal control print utility	Figure 186	O
DFHKC	Macro	Task service request	User	MCP
DFHKCD	CSECT	Dispatcher part of task control	Figure 7	S
DFHKCP	CSECT	Task control program	Figure 7	S
DFHKCPA	CSECT	Part of DFHKCP	Figure 7	S
DFHKCPB	CSECT	Part of DFHKCP	Figure 7	S

Name	Type	Description	Reference	Library
DFHKCRP	CSECT	Task control restart program		O
DFHKCSP	CSECT	Task SRB control program	Figure 7	
DFHKCSPS	DSECT	Task control static storage		S
DFHKCTWA	Macro	DFHKCP TWA description	System	
DFHKP	Macro	Keypoint service request	User	M
DFHKPBUF	DSECT	Keypoint buffer description	DFHKPP	M
DFHKPDDS	DSECT	Keypoint data definitions	DFHKPP	M
DFHKPPDS	DSECT	Keypoint program definitions/ keypoint control record	DFHKPP	M
DFHKPTDS	DSECT	Keypoint TCA record	DFHKPP	M
DFHKPTE	DSECT	TCTTE activity keypoint record	DFHKPP	M
DFHLFA	CSECT	LIFO prolog module	DFHCSA	M
DFHLFM	Macro	LIFO macro	System	
DFHLFO	CSECT	LIFO overflow module		S
DFHLFT	Macro	LIFO trace macro	System	M
DFHLLADS	DSECT	Load list area	DFHPCP	M
DFHLLDC	CSECT	Local logical device code	DFHTCT	M
DFHLUC	Macro	LU6.2 service request	System	M
DFHLUCDS	DSECT	LU6.2 control block		
DFHLUMDS	DSECT	LU6.2 control block, macro-level		S
DFHLUP	CSECT	LU6.2 services manager		S
DFHMAPDS	DSECT	BMS map description	DFHMCP	M
DFHMBO	Macro	Message backout table	DFHRUP	M
DFHMBODS	DSECT	Message backout table	DFHRUP	S
DFHMCAD	Macro	Map control area	DFHM32	M
DFHMCBDS	DSECT	BMS message control block	DFHMCP	M
DFHMCP	CSECT	Mapping control program		S
DFHMCPLK	Macro	Linkage to BMS modules	System	M
DFHMCRDS	DSECT	BMS message control record	DFHMCP	M
DFHMCX	CSECT	BMS fast path program	DFHMCP	S
DFHMDF	Macro	Generate BMS field definition	User	M
DFHMDI	Macro	Generate BMS map definition	User	M
DFHMGM	Macro	Message prototype macro	System	M
DFHMGMI0	Macro	Message prototype literal macro-1	System	M
DFHMGMI1	Macro	Message prototype literal macro-2	System	M
DFHMGP	CSECT	Error message writer		S
DFHMGT	CSECT	Message prototype control table		S
DFHMGT03	Source	Message prototype source	System	S
DFHMGT04	Source	Message prototype source	System	S
DFHMGT05	Source	Message prototype source	System	S
DFHMGT06	Source	Message prototype source	System	S
DFHMGT07	Source	Message prototype source	System	S
DFHMGT08	Source	Message prototype source	System	S
DFHMGT10	Source	Message prototype source	System	S
DFHMGT11	Source	Message prototype source	System	S
DFHMGT12	Source	Message prototype source	System	S
DFHMGT13	Source	Message prototype source	System	S
DFHMGT14	Source	Message prototype source	System	S
DFHMGT15	Source	Message prototype source	System	S
DFHMGT16	Source	Message prototype source	System	S
DFHMGT17	Source	Message prototype source	System	S
DFHMGT18	Source	Message prototype source	System	S
DFHMGT19	Source	Message prototype source	System	S
DFHMGT20	Source	Message prototype source	System	S

Name	Type	Description	Reference	Library
DFHMGT21	Source	Message prototype source	System	S
DFHMGT22	Source	Message prototype source	System	S
DFHMGT23	Source	Message prototype source	System	S
DFHMGT24	Source	Message prototype source	System	S
DFHMGT25	Source	Message prototype source	System	S
DFHMGT26	Source	Message prototype source	System	S
DFHMGT28	Source	Message prototype source	System	S
DFHMGT29	Source	Message prototype source	System	S
DFHMGT30	Source	Message prototype source	System	S
DFHMGT33	Source	Message prototype source	System	S
DFHMGT34	Source	Message prototype source	System	S
DFHMGT35	Source	Message prototype source	System	S
DFHMGT36	Source	Message prototype source	System	S
DFHMGT38	Source	Message prototype source	System	S
DFHMGT39	Source	Message prototype source	System	S
DFHMGT40	Source	Message prototype source	System	S
DFHMGT41	Source	Message prototype source	System	S
DFHMGT45	Source	Message prototype source	System	S
DFHMGT46	Source	Message prototype source	System	S
DFHMGT49	Source	Message prototype source	System	S
DFHMGT50	CSECT	PRDMP message text	DFHMGT	M
DFHMGT57	Source	Message prototype source	System	S
DFHMGT58	Source	Message prototype source	System	S
DFHMGT61	Source	Message prototype source	System	S
DFHMGT64	CSECT	Message generation table segment for XRF ZC		M
DFHMGT65	CSECT	Message generation table segment for XRF		M
DFHMGT66	CSECT	Message generation table segment for XRF		M
DFHMIN	Source	Input mapping	DFHMCE DFHM32	S
DFHMIR	CSECT	ISC request shipping - mirror program		S
DFHML1	CSECT	LU1 printer mapping program	DFHMCP	S
DFHMMM	DSECT	Term. definitions for autoinstall	System	O
DFHMRMC	DSECT	Message recovery message cache	DFHRUP	
DFHMSD	Macro	Generate BMS map set definition	User	M
DFHMSG	Macro	Message control program	DFHMCP	M
DFHMSGEN	Macro	Generates msgs. in BMS modules	System	M
DFHMSKM	Macro	LU6.2 security password	System	M
DFHMSP	CSECT	Message switching program	Figure 29	S
DFHMSPUT	Macro	Puts msgs. to terminals in BMS	System	M
DFHMTPA	CSECT	Master terminal program module A	Figure 27	S
DFHMTPB	CSECT	Master terminal program module B	DFHMTPA	S
DFHMTPC	CSECT	Master terminal program module C	DFHMTPA	S
DFHMTPD	CSECT	Master terminal program module D	DFHMTPA	S
DFHMTPE	CSECT	Master terminal program module E	DFHMTPA	S
DFHMTPF	CSECT	Master terminal program module F	DFHMTPA	S
DFHMTPG	CSECT	Master terminal program module G	DFHMTPA	S
DFHMTPUT	Macro	Puts BMS msgs. to master terminal	System	M
DFHMTTWA	Symbolic	Master terminal - common storage	DFHMTPA	M
DFHMTWM	Source	Master terminal - write message	DFHMTPA	M
DFHMXP	CSECT	Local queuing shipper	DFHMXP	S
DFHM32	CSECT	BMS 3270 mapping		S
DFHNLT	Macro	Nucleus load table	Table gen	M
DFHOC	Macro	Open/close service request	User	M
DFHOCLDS	DSECT	Open/close/locate parameter list	DFHOCP	M
DFHOCODS	DSECT	Open/close override	DFHOCP	M
DFHOCP	CSECT	Dynamic open/close program	Figure 28	O
DFHOSPWA	DSECT	BMS common control area	DFHMCP	M

Name	Type	Description	Reference	Library
DFHPAGE	DSECT	Paging request	System	M
DFHPAMDS	DSECT	Page allocation map	DFHSCP	S
DFHPBP	CSECT	BMS page and text build program	DFHMCP	S
DFHPC	Macro	Program service request	User	MCP
DFHPCEXT	Exit	Program control exit program	DFHPCP	S
DFHPCP	CSECT	Program control program	Figure 9	S
DFHPCRP	CSECT	Program control restart program	DFHPCP	O
DFHPCT	Macro	Program control table	Table gen	M
DFHPCTDS	DSECT	Program control table	System	
DFHPCTEN	Macro	Generate a PCT entry	DFHPCT	M
DFHPCTGP	Macro	Define all transactions with special properties	DFHPCT	M
DFHPDXRF	CSECT	PRDMP exit, process XRF blocks		O
DFHPDX1	CSECT	PRDMP exit		O
DFHPDX1A	CSECT	PRDMP exit	DFHPDX1	O
DFHPDX1B	CSECT	PRDMP exit	DFHPDX1	O
DFHPDX2	CSECT	PRDMP exit, CICS trace interpreter		O
DFHPEP	CSECT	Program error program		S
DFHPG	Macro	Punch job control	System gen	M
DFHPGADS	DSECT	BMS page control area	DFHMCP	M
DFHPGEN	Macro	Identify pregenerated modules	DFHSG	M
DFHPGENT	Macro	Create pregenerated module table	DFHPGEN	M
DFHPGLST	DSECT	Page fix/free request list	System	M
DFHPGR	Macro	CICS page fix	SVC	M
DFHPG2	Macro	SMP control card generator	DFHSG	M
DFHPG3	Macro	System generation inner macro	DFHSG	M
DFHPG4	Macro	System generation inner macro	DFHSG	M
DFHPHN	Source	Phonetic code conv. (offline)		S
DFHPHP	CSECT	Partition handling program	DFHMCP	O
DFHPLT	Macro	Program list table	Table gen	M
DFHPLTDS	DSECT	Program list table definition	DFHSTP	M
DFHPL1I	CSECT	PL/I interface	DFHPCP	S
DFHPL1OI	CSECT	PL/I optimizer interface	DFHPCP	S
DFHPPT	Macro	Processing program table	Table gen	M
DFHPPTDS	DSECT	Processing program table	DFHPCP	M
DFHPPTEN	Macro	Generate a PPT entry	DFHPPT	M
DFHPPTGP	Macro	Define all programs with special properties	DFHPPT	M
DFHPRADS	DSECT	Primed allocation map	DFHSCPA	M
DFHPRINT	Macro	DSECT print control	System	M
DFHPRK	CSECT	3270 hard copy support	DFHZCP	S
DFHPRMCK	Macro	Parameter checking macro	System	M
DFHPRPR	CSECT	HLL preprocessor		S
DFHPRV	Macro	PL/I 'F' library macro	System	M
DFHPSDDS	DSECT	Partition set control block		
DFHPSP	CSECT	System spooling interface program	Figure 176	O
DFHPSPDW	CSECT	System spooling interface, deferred work element processor		O
DFHPSPSS	CSECT	System spooling interface subtask		O
DFHPSPST	CSECT	System spooling interface control		O
DFHPSSVC	CSECT	System spooling interface, retrieve a data set name		O
DFHPUP	CSECT	Parameter utility program	DFHCSDUP	S
DFHPXR	CSECT	Post-exit routine		O
DFHP3270	Source	3270 print function support	DFHTCP	S

Name	Type	Description	Reference	Library
DFHQCADS	DSECT	Paging request	System	M
DFHQEADS	DSECT	Queue element area	User	M
DFHQRY	CSECT	Query transaction		O
DFHRCBP	CSECT	Recovery control restart program		S
DFHRCEX	CSECT	Recovery control enable exit		O
DFHRCP	CSECT	Recovery control program	Figure 174	O
DFHRE	Macro	Inner macro for DFHPPT and DFHPCT	Table gen	M
DFHRKB	CSECT	3270 hard copy support	DFHZCP	S
DFHRLR	CSECT	BMS route list resolution		S
DFHRMSY	CSECT	Task-related user exit resynchronization	DFHSPP	S
DFHRPCB	Macro	Extension to DL/I PCB control block - used to contain ISC information about PCB	System	M
DFHRSADS	DSECT	Register storage area	DFHPCP	M
DFHRTE	CSECT	Transaction routing program	Figure 193	S
DFHRUP	CSECT	Recovery utility program	Figure 35	S
DFHRWP70	CSECT	7770 read/write program	DFHTCP	S
DFHSAADS	DSECT	Storage accounting area	User	MCP
DFHSABDS	DSECT	Subsystem anchor block	System	S
DFHSC	Macro	Storage service request	User	MCP
DFHSCCOS	Symbolic	Storage control class of storage	DFHSCPA	M
DFHSCEXT	Exit	Storage control exit program	DFHSCP	S
DFHSCP	CSECT	Storage control program	Figure 8	S
DFHSCR	CSECT	Storage control recovery	DFHSCP	S
DFHSCTE	CSECT	Subsystem control table extension (interregion communication)	Anchor for IRC tables	
DFHSCXDS	DSECT	Extended storage anchor block		
DFHSDAM	CSECT	Direct-access logic module	DFHTDPA	
DFHSEC	Macro	Security operations macro	System	M
DFHSFP	CSECT	Sign-off program	DFHSNP	S
DFHSG	Macro	System generation	System gen	M
DFHSGA	Macro	System generation	System	M
DFHSIA1	CSECT	System init. - module A1	DFHSIP	S
DFHSIB1	CSECT	System init. - module B1	DFHSIP	S
DFHSICOM	Macro	System init. definitions	DFHSIP	M
DFHSIC1	CSECT	System init. - module C1	DFHSIP	S
DFHSID1	CSECT	System init. - module D1	DFHSIP	S
DFHSIDVA	Source	VSAM transient data initialization	DFHSID1	S
DFHSIE1	CSECT	System init. - module E1	DFHSIP	S
DFHSIF1	CSECT	System init. - module F1	DFHSIP	S
DFHSIG1	CSECT	System init. - module G1	DFHSIP	S
DFHSIH1	CSECT	System init. - module H1	DFHSIP	S
DFHSII1	CSECT	System init. - module I1	DFHSIP	S
DFHSIJ1	CSECT	System init. - module J1	DFHSIP	S
DFHSIP	CSECT	System initialization program	Figure 37	M
DFHSIPD	Macro	Generates initialization communication area	System	
DFHSIPDS	DSECT	SIP communication area	DFHSIP	S
DFHSIT	Macro	System initialization table	Table gen	M
DFHSK	Macro	Subtask management interface		M
DFHSKP	CSECT	Subtask management program	Figure 171	S
DFHSKR	Macro	Produce SKR table entries in SIT	Table gen	M
DFHSKRET	DSECT	Subtask management return codes	DFHSKP	S
DFHSLDC	CSECT	System logical device code table	DFHTCT	M
DFHSMPSG	Macro	Interprets SMP parameters	DFHSG	M
DFHSMPT	Macro	SMP control card generator	Table gen	M
DFHSNP	CSECT	Sign-on program		S
DFHSNT	Macro	Sign-on table	Table gen	M

Name	Type	Description	Reference	Library
DFHSORT	Macro	Sort auxiliary macro	Table gen	M
DFHSP	Macro	Sync point program request	User	MCP
DFHSPC	Source	LU6.2 sync point request	System	S
DFHSPP	CSECT	Sync point program		S
DFHSPZ	CSECT	Sync point resource manager		S
DFHSPZM	Macro	DFHSPZ interface	System	M
DFHSRADS	DSECT	CSA extension for HPO	System	M
DFHSRP	CSECT	System recovery program	Figure 32	S
DFHSRT	Macro	System recovery table	Table gen	M
DFHSRTDS	DSECT	System recovery table	DFHSRP	M
DFHSRXDS	DSECT	SRB and extensions in SQA	System	M
DFHSTAB	Macro	Table scan macro		M
DFHSTKC	CSECT	Supervisor statistics program		S
DFHSTLK	CSECT	Link statistics program		S
DFHSTP	CSECT	System termination program	Figure 41	S
DFHSTPD	CSECT	Supervisor statistics program	DFHSTKC	S
DFHSTR	Macro	Contains DSECT for DFHSRP's pgm. check and abend trace table	System	M
DFHSTSP	CSECT	Statistics summary control program		S
DFHSTTD	CSECT	Data management statistics program	DFHSTKC	S
DFHSTTR	CSECT	File and terminal statistics pgm.	DFHSTKC	S
DFHSTUP	CSECT	Statistics utility program	User	S
DFHSTUPD	CSECT	Statistics print routine		S
DFHSTUPO	CSECT	Statistics work space		S
DFHSTUPW	CSECT	Statistics print routine		S
DFHSYS	Macro	System definition macro	DFHVM	M
DFHTACLE	DSECT	TCT line entry prefix	DFHTCT	M
DFHTACP	CSECT	Terminal abnormal condition pgm.		S
DFHTAJP	CSECT	Time adjustment program		S
DFHTBO	Macro	Transaction backout table	DFHRUP	M
DFHTBODS	DSECT	Transaction backout table	DFHRUP	M
DFHTBS	Macro	Builder interface	System	O
DFHTBSB	CSECT	Add a node	DFHTBSxx	O
DFHTBSBP	CSECT	Recursive part of DFHTBSB	DFHTBSB	O
DFHTBSD	CSECT	Delete node program	DFHTBSxx	O
DFHTBSDP	CSECT	Recursive part of DFHTBSD	DFHTBSD	O
DFHTBSGB	DSECT	Builder services declarations	DFHTBSxx	O
DFHTBSL	CSECT	Create recovery record for node		O
DFHTBSLP	CSECT	Recursive part of DFHTBSL	DFHTBSL	O
DFHTBSQ	CSECT	Builder inquire process	DFHTBSxx	O
DFHTBSQP	CSECT	Recursive part of DFHTBSQ	DFHTBSQ	O
DFHTBSR	CSECT	Builder restore process	DFHTBSxx	O
DFHTBSRP	CSECT	Recursive part of DFHTBSR	DFHTBSR	O
DFHTBSSP	CSECT	Builder sync point processor	DFHTBSxx	O
DFHTBSOO	CSECT	Table builder services program	DFHTBSxx	O
DFHTC	Macro	Terminal service request	User	M
DFHTCA	Macro	Task control area (definition)	User	M
DFHTCADS	DSECT	Task control area (user)	User	MCP
DFHTCAM	Source	CICS - TCAM interface logic	DFHTCP	S
DFHTCBP	CSECT	Terminal control backout program	DFHTCP	S
DFHTCCBS	Source	Common bisync routine	DFHTCPA	S
DFHTCCLC	Source	Common line control logic	DFHTCP	S
DFHTCCOM	Source	Input data length computation	DFHTCP	S
DFHTCCSS	Source	Start-stop event analysis	DFHTCP	S

Name	Type	Description	Reference	Library
DFHTCDEF	Symbolic	Terminal control definitions	DFHTCP	S
DFHTCEXT	Exit	Terminal control exit routine	DFHTCP	S
DFHTCLIN	Macro	TCT TYPE=LINE generation	DFHTCT	
DFHTCMN1	Macro	TCT MNOTE generation	DFHTCT	
DFHTCMN2	Macro	TCT MNOTE generation	DFHTCT	
DFHTCMN3	Macro	TCT MNOTE generation	DFHTCT	
DFHTCMN4	Macro	TCT MNOTE generation	DFHTCT	
DFHTCMN5	Macro	TCT MNOTE generation	DFHTCT	
DFHTCMPV	Macro	TCT multipoint verification	DFHTCT	
DFHTCNBS	Source	Common nondial bisync routine	DFHTCP	S
DFHTCNED	Macro	TCT numeric editing	DFHTCT	
DFHTCN29	Source	2980 terminal dependent	DFHTCP	
DFHTCN36	Source	3600 BSC terminal dependent	DFHTCPA	S
DFHTCN70	Source	2770 terminal dependent	DFHTCP	S
DFHTCN74	Source	3740 terminal dependent	DFHTCP	S
DFHTCN80	Source	2780 terminal dependent	DFHTCP	S
DFHTCORS	Source	Terminal storage routine	DFHTCP	S
DFHTCP	CSECT	Terminal management program	Figure 15	S
DFHTCPCL	Macro	DFHZCP CALL macro	DFHZCP	M
DFHTCPCM	Macro	Common ZCP functions		M
DFHTCPQR	Macro	Queued response notification	System	M
DFHTCPRA	DSECT	Receive-any control element	DFHTCP	M
DFHTCPRT	Macro	DFHZCP RETURN macro	DFHZCP	M
DFHTCPSV	Macro	DFHZCP SAVE macro	DFHZCP	M
DFHTCPZR	Macro	Contains RPL extension for HPO	System	M
DFHTCQUE	Macro	DFHZCP QUEUE macro	DFHZCP	M
DFHTCRP	CSECT	Terminal control recovery program		O
DFHTCRPC	CSECT	XRF tracking interface for TCT contents		O
DFHTCRPG	CSECT	TCT recovery		
DFHTCRPL	CSECT	Install TCT macro definitions		
DFHTCRPS	CSECT	XRF tracking interface for ZCP sessions		O
DFHTCSAM	Source	Sequential terminal logic	DFHTCP	S
DFHTCSBS	Source	Switched-line bisync logic	DFHTCP	S
DFHTCSDS	Macro	TCT TYPE=SDSCI generation	DFHTCT	
DFHTCSIC	Source	Part of TCA	DFHTCADS	S
DFHTCSKC	Source	Part of TCA	DFHTCADS	S
DFHTCSNC	Source	Start-stop nonswitched logic	DFHTCP	S
DFHTCSSC	Source	Start-stop switched logic	DFHTCP	S
DFHTCS35	Source	3735 dial, terminal logic	DFHTCP	S
DFHTCS7N	Source	System/7 terminal logic	DFHTCP	S
DFHTCS70	Source	2770 dial logic	DFHTCP	S
DFHTCS74	Source	3740 dial logic	DFHTCP	S
DFHTCS80	Source	2780 dial logic	DFHTCP	S
DFHTCT	Macro	Terminal control table	Table gen	M
DFHTCTBD	Macro	TCT inner macro	DFHTCT	M
DFHTCTFN	Source	TCT TYPE=FINAL (VTAM)	DFHTCT	M
DFHTCTFX	DSECT	Terminal control table prefix	DFHTCP	M
DFHTCTG	Macro	TCT GPENTRY generation	DFHTCT	
DFHTCTI	Source	Task initiation logic	DFHTCP	
DFHTCTLC	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTLE	DSECT	TCT line entry	DFHTCT	M
DFHTCTLX	Source	Teletypewriter (WT) terminal logic	DFHTCPA	S
DFHTCTME	DSECT	TCT mode entry		
DFHTCTML	Macro	DFHTRMLST generation	DFHTCT	
DFHTCTNX	Macro	TCT inner macro	DFHTCT	M
DFHTCTPL	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTPX	Macro	Inner macro for DFHTCT	DFHTCT	M

Name	Type	Description	Reference	Library
DFHTCTRD	Macro	VTAM RDO builder	System	M
DFHTCTRE	Macro	TCT definition macro	System	M
DFHTCTRM	Macro	TCT TYPE=TERMINAL generation	DFHTCT	M
DFHTCTRN	Table	Terminal ctrl. translation tables	DFHTCP	S
DFHTCTSA	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTSB	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTSE	Macro	Generates ISC system entry	DFHTCT	M
DFHTCTSS	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTST	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTSV	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTTE	DSECT	TCT terminal entry	User	MCP
DFHTCTUA	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTUB	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTUV	Macro	Inner macro for DFHTCT	DFHTCT	M
DFHTCTWA	DSECT	TC transaction work area	DFHTCP	M
DFHTCTWX	Source	TWX terminal interface	DFHTCP	S
DFHTCTZE	Macro	TCTTE generation	Table gen	M
DFHTCT32	Macro	Inner macro for DFHTCT	DFHTCT	
DFHTCUIC	Source	Part of TCA	DFHTCADS	S
DFHTCUKC	Source	Part of TCA	DFHTCADS	S
DFHTC2KC	Source	Part of TCA	DFHTCADS	S
DFHTC3KC	Source	Part of TCA	DFHTCADS	S
DFHTC37D	Source	3740 dial logic	DFHTCP	S
DFHTC40N	Source	2740 nondial logic	DFHTCP	S
DFHTC40S	Source	2740 dial logic	DFHTCP	S
DFHTC41N	Source	2741 nondial logic	DFHTCP	S
DFHTC41S	Source	2741 dial logic	DFHTCP	S
DFHTC50N	Source	1050 nondial logic	DFHTCP	S
DFHTC50S	Source	1050 dial logic	DFHTCP	S
DFHTC70C	Source	3270 common logic	DFHTCP	S
DFHTC70L	Source	Local 3270 logic	DFHTCP	S
DFHTC70R	Source	Remote 3270 logic	DFHTCP	S
DFHTC77S	Source	7770 logic	DFHTCP	S
DFHTD	Macro	Transient data services request	User	MCP
DFHTDCI	DSECT	Map of VSAM CIDF	DFHTDP Figure 24	M
DFHTDDWP	Source	Transient data DWE processor	DFHTDP	S
DFHTDEXP	Source	Transient data extrapartition	DFHTDP	S
DFHTDEXT	Exit	Transient data exit routine	DFHTDP	S
DFHTDIA	DSECT	Transient data input area	User	MCP
DFHTDINP	Source	Transient data intrapartition	DFHTDP	S
DFHTDOA	DSECT	Transient data output area	User	MCP
DFHTDP	CSECT	Transient data control program	Figure 23	O
DFHTDQ	CSECT	Transient data cancel processor	DFHTDP	O
DFHTDR	CSECT	Transient data I/O error processor	DFHTDP	O
DFHTDRP	CSECT	Transient data phase 2 init.	DFHTDP	S
DFHTDSUB	Source	Transient data subroutines	DFHTDP	S
DFHTDX	CSECT	Transient data phase 1 init.	DFHTDP	O
DFHTEOF	CSECT	Tape end-of-file		S
DFHTEP	CSECT	Terminal error program	DFHTACP	S
DFHTEPA	Macro	Inner macro	DFHTEPM	M
DFHTEPC	Macro	Inner macro	DFHTEPM	M
DFHTEPM	Macro	TEP module generation	DFHTEP	M
DFHTEPS	Macro	Inner macro	DFHTEPM	M

Name	Type	Description	Reference	Library
DFHTEPT	Macro	TEP table generation	Table gen	M
DFHTERID	Symbolic	Terminal error definitions		M
DFHTIEDS	DSECT	Transaction interface element	DFHERM	S
DFHTIOA	DSECT	Terminal I/O area	User	S
DFHTLT	Macro	Terminal list table	Table gen	M
DFHTMDEL	DSECT	Table management directory element		
DFHTMDSG	DSECT	Table management directory segment		
DFHTMELD	DSECT	Table management lock block		
DFHTMP	CSECT	Table management program		0
DFHTMRQ	DSECT	Table management parameter list		
DFHTMSKT	DSECT	Table management hash table		
DFHTMSSA	DSECT	Table management static storage area		
DFHTOM	Macro	Terminal output macro instruction	User	M
DFHTON	CSECT	Terminal object resolution		0
DFHTORB				0
DFHTOR00	CSECT	Terminal object resolution		0
DFHTPE	DSECT	Terminal partition extension		
DFHTPP	CSECT	BMS terminal page processor		S
DFHTPQ	CSECT	BMS cleanup undelivered messages		S
DFHTPR	CSECT	BMS page retrieval		S
DFHTPS	CSECT	BMS terminal page scheduling		S
DFHTR	Macro	Trace service request	User	MCP
DFHTRACE	Macro	Trace system macro	System	M
DFHTRACK	Macro	Tracks CICS-DL/I interface	System	M
DFHTRADS	DSECT	Parameter list to DFHTRAP	System	S
DFHTRAP	CSECT	FE global trap/trace exit program	System	S
DFHTRP	CSECT	Trace control program	Figure 30	S
DFHTRPIO	CSECT	Trace control program - I/O section	DFHTRP	M
DFHTRZCP	CSECT	Terminal object builder		0
DFHTRZIP	CSECT	Session object builder		0
DFHTRZPP	CSECT	Pool object builder		0
DFHTRZXP	CSECT	Connection object builder		0
DFHTRZYB	CSECT	Type/bind matching routine		0
DFHTRZZP	CSECT	Terminal object matching		0
DFHTS	Macro	Temporary storage service request	User	MCP
DFHTSBP	CSECT	Temporary storage backout program		S
DFHTSCI	DSECT	Temporary storage record prefix	DFHTSP	M
DFHTSCTL	DSECT	Temporary storage control record	DFHTSP	M
DFHTSEXT	Exit	Temporary storage exit routine	DFHTSPA	S
DFHTSIOA	DSECT	Temporary storage I/O area	User	MCP
DFHTSMDS	DSECT	Temporary storage definitions	DFHTSP	M
DFHTSP	CSECT	Temporary storage control program	Figure 25	0
DFHTSRP	CSECT	Temporary storage recovery program	DFHRUP	S
DFHTST	Macro	Temporary storage table	DFHTSP	M
DFHTSTDS	DSECT	Maps entry in temporary storage table	DFHTSP	M
DFHTTPDS	DSECT	BMS - terminal type parameter	DFHMCP	M
DFHTUL	DSECT	Standard-labeled tapes formats	DFHVCP	S
DFHTUP	CSECT	Trace utility program		S
DFHTURDDS	DSECT	Unit of recovery descriptor		S
DFHTUREN	CSECT	Trace table decoding routine	DFHTUP	S
DFHTUTEN	Macro	Trace table generation macro	DFHTUREN	M
DFHTXRP	CSECT	Install macro-defined TCTTEs		S
DFHTXRPR	CSECT	Recursive routine for DFHTXRP		S

Name	Type	Description	Reference	Library
DFHTZEXT	CSECT	DFHZCP user exit	DFHZCP	S
DFHUEH	CSECT	User exit handler	Figure 182	S
DFHUEM	CSECT	User exit manager	Figure 182	S
DFHUEXIT	Macro	For generating user-exit-dependent code	System	M
DFHUEXPT	Macro	Inner macro containing user exit definitions	DFHUEXIT	M
DFHURDDS	DSECT	Unit of recovery descriptor		
DFHURLDS	DSECT	BMS - user-supplied route list	User	MCP
DFHUSBP	CSECT	User backout program		S
DFHUTDEV	CSECT	Offline utility device type determination subroutine	DFHDUP DFHSTUP	
DFHVAP	CSECT	VSAM subtask attach and wait	DFHFCP	S
DFHVC	Macro	Volume management request	DFHVCP	M
DFHVCP	CSECT	Volume control program	DFHVCP	O
DFHVCPDY	CSECT	Volume control dummy program	DFHVCP	S
DFHVM	Macro	Version/modification level gen	System gen	M
DFHVOLDS	DSECT	Standard-labeled tape area	DFHVCP	M
DFHVSA	DSECT	VSAM subtask area		
DFHVSP	CSECT	VSAM subtask program	DFHFCP	O
DFHVSWA	DSECT	VSAM work area	User	MCP
DFHVTWA	Macro	NACP/NEP transaction work area	System	S
DFHWCCS	CSECT	CAVM common services		O
DFHWCGNT	CSECT	CAVM entry point table for routines above 16-megabyte line		O
DFHWCSNT	CSECT	CAVM entry point table for routines below 16-megabyte line		O
DFHWDATT	CSECT	XRF process dispatcher attach control		O
DFHWDINA	CSECT	XRF process dispatcher initialization		O
DFHWDISP	CSECT	XRF process dispatcher		O
DFHWDSRP	CSECT	PC/ABEND handler for XRF dispatcher		O
DFHWDWAT	CSECT	XRF process dispatcher wait services		O
DFHWKP	CSECT	Warm keypoint program	Figure 171	O
DFHWLF	Macro	XRF LIFO free storage request	System	O
DFHWLFRE	CSECT	XRF LIFO free allocation service		O
DFHWLG	Macro	XRF LIFO get storage request	System	O
DFHWLGET	CSECT	XRF LIFO get allocation service		O
DFHWMCAN	CSECT	XRF message manager, DFHKCP cancel exit routine		O
DFHWMG1	CSECT	XRF message manager, GETMSG process		O
DFHWMI	CSECT	XRF message manager, sign on initialization routine		O
DFHWMMT	CSECT	XRF message manager, I/O services		O
DFHWMPG	CSECT	XRF message manager, data copying service		O
DFHWMP1	CSECT	XRF message manager, PUTMSG process		O
DFHWMQG	CSECT	XRF message manager, CICS TCB part of GETMSG processing		O
DFHWMQH	CSECT	XRF message manager, message block services for GETMSG		O
DFHWMQP	CSECT	XRF message manager, CICS TCB part of PUTMSG processing		O
DFHWMQS	CSECT	XRF message manager, work queue services		O

Name	Type	Description	Reference	Library
DFHWMRD	CSECT	XRF message manager, message reader		O
DFHWMR1	CSECT	XRF message manager, PUTREQ/PUTRSP process		O
DFHWMS	CSECT	XRF message manager, request interface		O
DFHWMS10	CSECT	XRF message manager, environment switch routine		O
DFHWMS20	CSECT	XRF message manager, request router		O
DFHWMWR	CSECT	XRF message manager, output routine		O
DFHWNFDS	DSECT	CAVM NOTIFY exit parameter block		O
DFHWOS	CSECT	Overseer bootstrap module		O
DFHWOSA	CSECT	Overseer initialization module		O
DFHWOSB	CSECT	Overseer services module		O
DFHWOSM	Macro	Overseer interface definition		M
DFHWSMDS	DSECT	CAVM storage management record		O
DFHWSRTR	CSECT	CAVM state management request router and subtask entry point		O
DFHWSSDS				O
DFHWSSN1	CSECT	CAVM state management sign-on initial entry point		O
DFHWSSN2	CSECT	CAVM state management sign-on request handler		O
DFHWSSN3	CSECT	CAVM state management data set initialization routine		O
DFHWSSOF	CSECT	CAVM state management sign-off request handler		O
DFHWSSR	CSECT	CAVM surveillance status reader		O
DFHWSSW	CSECT	CAVM surveillance status writer		O
DFHWSTI	CSECT	CAVM surveillance tick generator and system status monitor		O
DFHWSTKV	CSECT	CAVM state management takeover request handler		O
DFHWSXDS	DSECT	NOTIFY exit control block		O
DFHWSXPI	CSECT	CAVM state management CAVM process initialization		O
DFHWTADS	DSECT	Takeover initiation program arguments	System	O
DFHWTI	CSECT	Takeover initiation program	System	O
DFHWTIC	CSECT	Takeover initiation program - CLT accessing	DFHWTI	O
DFHWTII	CSECT	Takeover initiation program - inquire job status	DFHWTI	O
DFHWTIJ	CSECT	Takeover initiation program - job termination/wait	DFHWTI	O
DFHWTO	Macro	Write to console operator	System	M
DFHWTRP	CSECT	XRF trace routine		O
DFHXFDL	Macro	DL/I function request shipping	DFHXFP	M
DFHXFFC	Macro	FC function request shipping	DFHXFP	M
DFHXFHED	Macro	Macro to produce headings for transformation program	DFHXFP	M
DFHXFIC	Macro	IC function request shipping	DFHXFP	M
DFHXFIOA	DSECT	Transformer I/O area	System	M
DFHXFJC	Macro	JC function request shipping	DFHXFP	M
DFHXFMOD	Macro	Macro to produce transformation programs	DFHXFP	M

Name	Type	Description	Reference	Library
DFHXFP	CSECT	Online transformation flow program	DFHXFP	S
DFHXFQ	CSECT	Batch data transformation program		S
DFHXFQU	Macro	TD and TS function request shipping	DFHXFP	M
DFHXFSTG	Macro	XF control block and transformer	DFHXFP	M
DFHXFX	CSECT	Optimized transformer program		S
DFHXJCC	CSECT	User-replaceable journal close	Figure 183	O
DFHXJCO	CSECT	User-replaceable journal open	Figure 183	O
DFHXLT	Macro	Transaction list table	Table gen	M
DFHXLTDS	DSECT	Transaction list table	System	M
DFHXMP	CSECT	Cross-memory program		S
DFHXMSG	CSECT	Default XRF recovery message	DFHZNAC	O
DFHXR	Macro	XRF code generation macro		M
DFHXRA	CSECT	XRF request processing program		O
DFHXRB	CSECT	XRF NOTIFY exit program		O
DFHXRC	CSECT	XRF inquire status exit program		O
DFHXRCP	CSECT	XRF console communication program		O
DFHXRD	CSECT	XRF bootstrap program		O
DFHXRE	CSECT	XRF bootstrap program		O
DFHXRF	CSECT	XRF bootstrap program		O
DFHXRHDS	DSECT	XRF status data definition		M
DFHXRSDS	DSECT	XRF static storage definition		M
DFHXRSP	CSECT	XRF surveillance program		O
DFHXSE	CSECT	Security external checking program		S
DFHXSP	CSECT	Security program	DFHSEC	S
DFHXSS	CSECT	RACF module interface	DFHCSVC	S
DFHXTAB	Macro	Inner macro	DFHTCTST	MCP
DFHXTCI	Source	XRF terminal switching	System	O
DFHXTP	CSECT	Terminal sharing transformation program	Figure 193	
DFHXTSTG	Macro	XTP parameter list system	System	M
DFHZABD	CSECT	No VTAM support module		
DFHZACT	CSECT	Activate chain	DFHZCP	S
DFHZAIT	CSECT	Attach initialization table	DFHZCP	S

Name	Type	Description	Reference	Library
DFHZAND	CSECT	Abend control block		
DFHZARL	CSECT	LU6.2 request process program	DFHZCC	S
DFHZARM	CSECT	LU6.2 macro requests program	DFHZCC	S
DFHZARQ	CSECT	Application request	DFHZCP	S
DFHZASX	CSECT	DFASY exit	DFHZCP	M
DFHZATD	CSECT	Automatic terminal installation		O
DFHZATDX	CSECT	Automatic terminal installation user program		O
DFHZATI	CSECT	Automatic task initiation	DFHZCP	S
DFHZATT	CSECT	Task attach	DFHZCP	S
DFHZBAN	CSECT	Terminal control bind analysis		O
DFHZBKT	CSECT	LU6.2 bracket state program	DFHZCC	S
DFHZBSM	Macro	LU6.2 bracket state macro	DFHZCC	M
DFHZCA	CSECT	VTAM working set module	Figure 199	S
DFHZCB	CSECT	VTAM working set module	Figure 199	S
DFHZCC	CSECT	VTAM working set module	Figure 199	S
DFHZCHS	CSECT	LU6.2 chain state program	DFHZCC	S
DFHZCHM	Macro	LU6.2 chain state macro	DFHZCC	M
DFHZCLS	CSECT	CLSDST request	DFHZCP	S
DFHZCLX	CSECT	CLSDST exit	DFHZCP	S
DFHZCNA	CSECT	System console activity control	DFHZCP	
DFHZCNR	CSECT	System console applic request	DFHZCX	
DFHZCNT	CSECT	LU6.2 contention state program	DFHZCC	S
DFHZCNM	Macro	LU6.2 contention state macro	DFHZCC	M
DFHZCP	CSECT	Terminal management program	Figure 15	S
DFHZCPBK	Macro	Bracket control	System	M
DFHZCQ	Macro	Terminal control install interface	System	O
DFHZCQDL	CSECT	Dynamic delete TCT element	DFHZNAC	O
DFHZCQIF	DSECT	RDO and DFHZCQ table	System	O
DFHZCQIN	CSECT	Initialize DFHZCQ	DFHTCRP	O
DFHZCQIQ	CSECT	Inquire about TCT entry	DFHZTSP	O
DFHZCQIS	CSECT	Install a TCTTE		O
DFHZCQIT	CSECT	Add a macro-generated TCTTE		O
DFHZCQPS	DSECT	Builder work space		O
DFHZCQRC	CSECT	Merge terminal control definitions		O
DFHZCQRS	CSECT	Restore a terminal control resource		O
DFHZCQ00	CSECT	Dynamic add/replace TCT elements		O
DFHZCRQ	CSECT	CTYPE command request	DFHZCP	S
DFHZCW	CSECT	VTAM nonworking set module	Figure 199	S
DFHZCX	CSECT	LOCATE, ISC/IRC request		S
DFHZCY	CSECT	VTAM nonworking set module	Figure 199	S
DFHZCZ	CSECT	VTAM nonworking set module	Figure 199	S
DFHZDET	CSECT	Task detach	DFHZCP	S
DFHZDSP	CSECT	Dispatcher	DFHZCP	S
DFHZDST	CSECT	SNA-ASCII translator	DFHZCP	S
DFHZDWE	CSECT	DWE processing	DFHZCP	S
DFHZEMW	CSECT	Error message writer	DFHZCP	S
DFHZEPD	DSECT	TCP/ZCP entry address list	DFHZCP	M
DFHZEQU	Source	ZCP equates	DFHZCP	M
DFHZERH	CSECT	LU6.2 error program	DFHZCP	S
DFHZERRM	Macro	ZCP error-handling macro	System	M
DFHZEV1	CSECT	LU6.2 security encryption program part 1		S

Name	Type	Description	Reference	Library
DFHZEV2	CSECT	LU6.2 security encryption program part 2		S
DFHZFRE	CSECT	FREEMAIN request	DFHZCP	S
DFHZGET	CSECT	GETMAIN request	DFHZCP	S
DFHZHPCH	Macro	Generates authorized path CHECK or CHECK macro	DFHZCP	M
DFHZHPRV	Macro	Generates authorized path RECEIVE or RECEIVED macro	DFHZCP	M
DFHZHPRX	CSECT	Authorized path SRB mode VTAM EXECRPL	DFHZCP	S
DFHZHPSD	Macro	Generates authorized path SEND or SEND macro	DFHZCP	M
DFHZHPSR	CSECT	Authorized path SRB register	DFHZCP	S
DFHZIRCT	Macro	Build FMH macro	System	M
DFHZISP	CSECT	Allocate/point/free	DFHZCP	S
DFHZIS1	CSECT	Prepare/SPR/commit/abort	DFHZCP	S
DFHZIS2	CSECT	IRC internal requests	DFHZCP	S
DFHZLEX	CSECT	LERAD exit	DFHZCP	M
DFHZLGX	CSECT	Logon exit	DFHZCP	M
DFHZLOC	CSECT	Locate	DFHZCP	S
DFHZLRP	CSECT	Logical record presentation	DFHZCP	S
DFHZLTX	CSECT	LOSTERM exit	DFHZCP	M
DFHZLUS	CSECT	LU6.2 session management program	DFHZCP	S
DFHZMJA	Source	NACP sense code table generation	DFHZCP	M
DFHZMJM	Macro	NACP sense code table generation macro	DFHZCP	M
DFHZNAC	CSECT	NACP	DFHZCP	S
DFHZNCA	Source	NACP message table generation	DFHZCP	M
DFHZNCE	Source	NACP interval equates	DFHZCP	M
DFHZNCM	Macro	NACP message table generation macro	DFHZCP	M
DFHZNEP	CSECT	Node error program	DFHZNEPI	S
DFHZNEPI	Macro	NEP interface generation	User	M
DFHZNSP	CSECT	VTAM services procedure error exit	DFHZCP	S
DFHZOPA	CSECT	Dynamic VTAM open	DFHZCP	S
DFHZOPN	CSECT	OPNDST	DFHZCP	S
DFHZOPX	CSECT	OPNDST exit	DFHZCP	S
DFHZQUE	CSECT	Attach chain and queue subroutine	DFHZCP	S
DFHZRAC	CSECT	Receive-any completion	DFHZCP	S
DFHZRAQ	CSECT	Read ahead queuing	DFHZCP	S
DFHZRAR	CSECT	Read ahead retrieval	DFHZCP	S
DFHZRLG	CSECT	Response logger	DFHZCP	S
DFHZRLX	CSECT	LU6.2 receive exit program	DFHZCC	S
DFHZRRX	CSECT	Release request exit	DFHZCP	M
DFHZRSP	CSECT	Resync send program	DFHZCP	S
DFHZRST	CSECT	RESETR	DFHZCP	S
DFHZRSY	CSECT	Resynchronization	DFHZCP	S
DFHZRTTE	Macro	Resource types table	System	M
DFHZRVL	CSECT	LU6.2 receive program	DFHZCC	S
DFHZRVS	CSECT	Receive specific	DFHZCP	S
DFHZRVX	CSECT	Receive specific exit	DFHZCP	S
DFHZSAX	CSECT	Send DFASY exit	DFHZCP	S
DFHZSCX	CSECT	Session control input exit	DFHZCP	M
DFHZSDA	CSECT	Send asynchronous command	DFHZCP	S
DFHZSDL	CSECT	LU6.2 send program	DFHZCP	S

Name	Type	Description	Reference	Library
DFHZSDR	CSECT	Send response	DFHZCP	S
DFHZSDS	CSECT	Send DFSYN	DFHZCP	S
DFHZSDX	CSECT	Send DFSYN data exit	DFHZCP	S
DFHZSES	CSECT	SESSIONC	DFHZCP	S
DFHZSEX	CSECT	SESSIONC exit	DFHZCP	S
DFHZSHU	CSECT	Checks shutdown status for VTAM terminals	DFHZCP	
DFHZSIM	CSECT	SIMLOGON	DFHZCP	S
DFHZSIX	CSECT	SIMLOGON exit	DFHZCP	M
DFHZSKR	CSECT	Command response	DFHZCP	S
DFHZSLS	CSECT	Set logon start	DFHZCP	S
DFHZSLX	CSECT	LU6.2 send exit program	DFHZCC	S
DFHZSSX	CSECT	Send DFSYN exit	DFHZCP	S
DFHZSTU	CSECT	Terminal control status change		
DFHZSUP	CSECT	Startup task	DFHZCP	S
DFHZSYN	CSECT	VTAM recovery module		
DFHZSYX	CSECT	SYNAD exit	DFHZCP	M
DFHZTAX	CSECT	Turnaround exit	DFHZCP	M
DFHZTPX	CSECT	TPEND exit	DFHZCP	M
DFHZTR	Macro	Trace macro (ZCP)	DFHZCP	M
DFHZTRA	CSECT	VTAM trace module		
DFHZTSP	CSECT	Transaction routing program		
DFHZUCT	CSECT	Upper case translate	DFHZCP	S
DFHZUIX	CSECT	User input exit	DFHZCP	S
DFHZUSR	CSECT	LU6.2 conversation state program	DFHZCC	S
DFHZUSRM	Macro	LU6.2 conversation state macro	DFHZCC	M
DFHZUTM	Macro	User-ID table manager macro	System	M
DFHZXCU	CSECT	VTAM XRF catch-up program		O
DFHZXQO	CSECT	XRF ZCP tracking queue organizer		O
DFHZXQOS	Symbolic	DFHZCQO internal control blocks		M
DFHZXRC	CSECT	XRF recovery	DFHZCP	S
DFHZXRE0	CSECT	VTAM reconnect transaction		
DFHZXRPL	Macro	Clear RPL	DFHZCP	M
DFHZXS	Macro	Interface to DFHZXST		M
DFHZXST	CSECT	XRF ZCP session-state tracking		O
DFHZXSTS	CSECT	SETLOGON routine		O
DFH2980	Symbolic	Special characters for 2980	User	CP
DLIUIB	Macro	DL/I user interface block	User	S
DLIUIB	DSECT	DL/I user interface block	User	CP
DLMT1	DSECT	CEMT parameter list	System	M
DLMT2	DSECT	CEMT parameter list	System	M
SPHEADER	DSECT	Subpool header	DFHSCP	
TCTTETTE	DSECT	Terminal table extension	DFHTCP	
UD	DSECT	Updated data base element	System	M
WTRETWA	DSECT	Weighted retrieval work area	DFHBFP	

Name	Type	Description	Reference	Library
The following modules are related to sample programs:				
DFHCICSU	Sample	CICS startup	OPG	Y
DFHCLT1$	Sample	Sample command list table		X
DFHDCT2$	Sample	Destination control table	OPG	X
DFHFCT2$	Sample	File control table	OPG	X
DFHIVMBT	Sample	Verify CICS batch with monitoring	OPG	I
DFHIVMOL	Sample	Verify CICS online with monitoring	OPG	I
DFHIVPBT	Sample	Verify CICS batch	OPG	I
DFHIVPOL	Sample	Verify CICS online	OPG	I
DFHIVXRO	Sample			X
DFHJCT2$	Sample	Journal control table	OPG	X
DFHJCT7$	Sample			X
DFHMCT2$	Sample	Monitoring control table	OPG	X
DFHMDLSG	Sample	Input to assemble DL/I interface programs	OPG	X
DFHPCT1$	Sample	Program control table	OPG	X
DFHPPT1$	Sample	Processing program table	OPG	X
DFHRTY	Sample	Retry program		S
DFHSAMP	Sample	Application program for 3770 batch LU		S
DFHSDR$	Sample	BTAM error statistics recorder	OPG	X
DFHSIT6$	Sample	System initialization table	OPG	X
DFHSIT7$	Sample			X
DFHSNEP	Macro	NEP generation macro	System gen	M
DFHSNEPH	Macro	NEP inner macro	DFHSNEP	M
DFHSNET	Macro	NEP generation macro	Table gen	M
DFHSNT1$	Sample	Sign-on table	OPG	X
DFHSPHCP	Sample	3790 program	OPG	S
DFHSPLZ1	Source	NCP/VS and VTAM definitions for 3600		S
DFHSPLZ2	Source	TCT for 3600, 3650, 3790		S
DFHSPLZ3	Source	3601 configuration		S
DFHSPL14	Sample	Program to control 3614		S
DFHSPPC0	Sample	3770 PC program		S
DFHSPTM1	Source	TCAM MCP and message handlers	CG	X
DFHSPTM2	Source	TCAM MCP for TCAM direct and tables	CG	X
DFHSP14A	Sample	Program to control 3614		S
DFHSP14B	Sample	3614 transaction processing program		S
DFHSP36B	Source	3600 bisync application program and tables		S
DFHSRT1$	Sample	System recovery table	OPG	X
DFHSVCSG	Sample	Input to assemble programs dependent on CICSSVC and SRBSVC	OPG	X
DFHTCT5$	Sample	Terminal control table	OPG	X
DFHZAPB	Sample	3770 application program		S
DFHZCTDX	Sample	Autoinstall user program - COBOL	CG	X
DFHZPTDX	Sample	Autoinstall user program - PL/I	CG	X
DFH99BC	Sample	Dynamic allocation - convert to binary target	CG	X
DFH99BLD	Other	Dyn alloc - JCL to build prog	CG	I
DFH99CC	Sample	Dyn alloc - char and num string conversion	CG	X

Name	Type	Description	Reference	Library
DFH99DY	Sample	Dyn alloc - issue SVC and analyze	CG	X
DFH99FP	Sample	Dyn alloc - process function keyword	CG	X
DFH99GI	Sample	Dyn alloc - format display and get input	CG	X
DFH99KC	Sample	Dyn alloc - keyword value conversion	CG	X
DFH99KH	Sample	Dyn alloc - list keywords for help	CG	X
DFH99KO	Sample	Dyn alloc - process operator keywords	CG	X
DFH99KR	Sample	Dyn alloc - convert returned value to keyword	CG	X
DFH99LK	Sample	Dyn alloc - search keyset for given token	CG	X
DFH99M	Sample	Dyn alloc - macro		M
DFH99ML	Sample	Dyn alloc - build msg. text from token list	CG	X
DFH99MM	Sample	Dyn alloc - main control program	CG	X
DFH99MP	Sample	Dyn alloc - message filing routine	CG	X
DFH99MT	Sample	Dyn alloc - match abbreviation with keyword	CG	X
DFH99RP	Sample	Dyn alloc - process returned values	CG	X
DFH99T	Sample	Dyn alloc - table of keywords	CG	X
DFH99TK	Sample	Dyn alloc - tokenize input command	CG	X
DFH99TX	Sample	Dyn alloc - text display routine	CG	X
DFH99VH	Sample	Dyn alloc - list description for help	CG	X
DFH$AALL	Sample	Inquiry/update	APRM(C)	X
DFH$ABRW	Sample	Browse	APRM(C)	X
DFH$ACOM	Sample	Order entry queue print	APRM(C)	X
DFH$AFIL	Sample	Customer file (FILEA) record layout	APRM(C)	X
DFH$AGA	Sample	Generated version of DFH$AMA	APRM(C)	S
DFH$AGB	Sample	Generated version of DFH$AMB	APRM(C)	S
DFH$AGC	Sample	Generated version of DFH$AMC	APRM(C)	S
DFH$AGD	Sample	Generated version of DFH$AMD	APRM(C)	S
DFH$AGK	Sample	Generated version of DFH$AMK	APRM(C)	S
DFH$AGL	Sample	Generated version of DFH$AML	APRM(C)	S
DFH$ALOG	Sample	Audit trail (log) record layout	APRM(C)	X
DFH$AL86	Sample	Order entry queue record layout	APRM(C)	X
DFH$AMA	Sample	Operator instructions map source	APRM(C)	X
DFH$AMB	Sample	Customer details map source	APRM(C)	X
DFH$AMC	Sample	File browse map source	APRM(C)	X
DFH$AMD	Sample	Low balance inquiry map source	APRM(C)	X
DFH$AMK	Sample	Order entry map source	APRM(C)	X
DFH$AML	Sample	Order report map source	APRM(C)	X
DFH$AMNU	Sample	Operator instructions	APRM(C)	X
DFH$AREN	Sample	Order entry	APRM(C)	X
DFH$AREP	Sample	Low balance inquiry	APRM(C)	X
DFH$ATMS	Macro	Dummy code point for ATMS	System gen	M
DFH$AXRO	Sample			X
DFH$BTCH	Sample	Batch test data for DFHIVPBT	OPG	X
DFH$BTMN	Sample	Batch test data for DFHIVPBT and DFHCMF	OPG	X
DFH$CALL	Sample	Inquiry/update	APRM(C)	X
DFH$CBRW	Sample	Browse	APRM(C)	X
DFH$CCOM	Sample	Order entry queue print	APRM(C)	X
DFH$CFIL	Sample	Customer file (FILEA) record layout	APRM(C)	C
DFH$CLOG	Sample	Audit trail (log) record layout	APRM(C)	C
DFH$CL86	Sample	Order entry queue record layout	APRM(C)	C

Name	Type	Description	Reference	Library
DFH$CMA	Sample	Operator instructions map source	APRM(C)	X
DFH$CMB	Sample	Customer details map source	APRM(C)	X
DFH$CMC	Sample	File browse map source	APRM(C)	X
DFH$CMD	Sample	Low balance inquiry map source	APRM(C)	X
DFH$CMK	Sample	Order entry map source	APRM(C)	X
DFH$CML	Sample	Order report map source	APRM(C)	X
DFH$CMNU	Sample	Operator instructions	APRM(C)	X
DFH$CMP	Sample	Mapset - COBOL	APRM(C)	X
DFH$CPKO	Sample	Keystroke overlap - COBOL	APRM(C)	X
DFH$CPLA	Sample	Look-aside query - COBOL	APRM(C)	X
DFH$CREN	Sample	Order entry	APRM(C)	X
DFH$CREP	Sample	Low balance inquiry	APRM(C)	X
DFH$DCTD	Sample	DCT SDSCI entries	OPG	X
DFH$DCTR	Sample	DCT basic facilities	OPG	X
DFH$DCTS	Sample	DCT sample programs	OPG	X
DFH$FAIN	Sample	Data for batch load of FILEA	OPG	X
DFH$FCTR	Sample	FCT entry for DFHCSD	OPG	X
DFH$FCTS	Sample	FCT entry for sample FILEA	OPG	X
DFH$ICIC	Sample	CICS-CICS/IMS/VS conversation	ICG	X
DFH$IFBL	Sample	Remote file browse - local	ICG	X
DFH$IFBR	Sample	Remote file browse - remote	ICG	X
DFH$IGB	Sample	Generated version of DFH$IMB	APRM(C)	S
DFH$IGC	Sample	Generated version of DFH$IMC	APRM(C)	S
DFH$IGS	Sample	Generated version of DFH$IMS	APRM(C)	S
DFH$IGX	Sample	Generated version of DFH$IMX	APRM(C)	S
DFH$IG1	Sample	Generated version of DFH$IM1	APRM(C)	S
DFH$IG2	Sample	Generated version of DFH$IM2	APRM(C)	S
DFH$IMB	Sample	Remote file browse - mapset	ICG	X
DFH$IMC	Sample	CICS-CICS/IMS/VS conversation - mapset	ICG	X
DFH$IMS	Sample	CICS-IMS/VS conversation/DPO - mapset	ICG	X
DFH$IMSN	Sample	CICS-IMS/VS conversation	ICG	X
DFH$IMSO	Sample	CICS-IMS/VS demand paged output	ICG	X
DFH$IMX	Sample	Local to remote temporary storage queue transfer - mapset	ICG ICG	X
DFH$IM1	Sample	Temporary storage record retrieval - mapset 1	ICG	X
DFH$IM2	Sample	Temporary storage record retrieval - mapset 2	ICG	X
DFH$IQRD	Sample	Temporary storage record retrieval - local display	ICG	X
DFH$IQRL	Sample	Temporary storage record retrieval - local request	ICG	X
DFH$IQRR	Sample	Temporary storage record retrieval - remote request	ICG	X
DFH$IQXL	Sample	Local to remote temporary storage queue transfer - local	ICG	X
DFH$IQXR	Sample	Local to remote temporary storage queue transfer - remote	ICG	X
DFH$JCT1	Sample	JCT journal 01 - system log	OPG	X
DFH$JCT2	Sample	JCT journal 02 - user journal for monitoring	OPG	X
DFH$JCT3	Sample			X
DFH$LDSP	Sample	Create FILEA data file	OPG	X
DFH$MOLS	Sample	Monitor listing	CG	X
DFH$PALL	Sample	Inquiry/update	APRM(C)	X
DFH$PBRW	Sample	Browse	APRM(C)	X
DFH$PCOM	Sample	Order entry queue print	APRM(C)	X

Name	Type	Description	Reference	Library
DFH$PCTA	Sample	PCT entries for assembler samples	OPG	X
DFH$PCTC	Sample	PCT entries for COBOL samples	OPG	X
DFH$PCTP	Sample	PCT entries for PL/I samples	OPG	X
DFH$PCTR	Sample	PCT required entries	OPG	X
DFH$PFIL	Sample	Customer file (FILEA) record layout	APRM(C)	P
DFH$PLOG	Sample	Audit trail (log) record layout	APRM(C)	P
DFH$PL86	Sample	Order entry queue record layout	APRM(C)	P
DFH$PMA	Sample	Operator instructions map source	APRM(C)	X
DFH$PMB	Sample	Customer details map source	APRM(C)	X
DFH$PMC	Sample	File browse map source	APRM(C)	X
DFH$PMD	Sample	Low balance inquiry map source	APRM(C)	X
DFH$PMK	Sample	Order entry map source	APRM(C)	X
DFH$PML	Sample	Order report map source	APRM(C)	X
DFH$PMNU	Sample	Operator instructions	APRM(C)	X
DFH$PMP	Sample	Mapset - PL/I	APRM(C)	X
DFH$PPKO	Sample	Keystroke - PL/I	APRM(C)	X
DFH$PPLA	Sample	Look-aside query - PL/I	APRM(C)	X
DFH$PPTA	Sample	PPT entries for assembler samples	OPG	X
DFH$PPTC	Sample	PPT entries for COBOL samples	OPG	X
DFH$PPTP	Sample	PPT entries for PL/I samples	OPG	X
DFH$PPTR	Sample	PPT required entries	OPG	X
DFH$PREN	Sample	Order entry	APRM(C)	X
DFH$PREP	Sample	Low balance inquiry	APRM(C)	X
DFH$PS	Sample	Partition set	APRM(C)	X
DFH$RACF	Sample	RACF class descriptor table	APRM(C)	X
DFH$SET	Sample	ACCTSET - map set source	OPG	X
DFH$SINX	Sample	ACCTINDX - batch index file recovery	OPG	X
DFH$SIXR	Sample	ACIXREC - index record	OPG	C
DFH$SREC	Sample	ACCTREC - account record	OPG	C
DFH$S00	Sample	ACCT00 - menu display	OPG	X
DFH$S01	Sample	ACCT01 - initial request processing	OPG	X
DFH$S02	Sample	ACCT02 - update processing	OPG	X
DFH$S03	Sample	ACCT03 - requests for printing	OPG	X
DFH$S04	Sample	ACCT04 - error processing	OPG	X
DFH$TCTB	Sample	TCT BTAM local and remote	OPG	X
DFH$TCTC	Sample	TCT console	OPG	X
DFH$TCTS	Sample	TCT sequential (CRLP)	OPG	X
DFH$TCTV	Sample	TCT VTAM SNA and non-SNA	OPG	X
DFH$TDWT	Sample	Transient data write	OPG	X
DFH$XRDS	Sample			X

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Numerics

Customer Information Control System
CICS/MVS Version 2 Release 1
Diagnosis Reference

READER'S COMMENT FORM

Order No. LC33-0517-0

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative, or the IBM branch office serving your locality.

Number of latest Technical Newsletter for this publication...

Note: *Staples can cause problems with automated mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.*

If you want an acknowledgement, give your name and address below.

Name .....

Job Title ..... Company .....

Address .....

..... Zip .....

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

LC33-0517-0

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

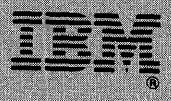
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1H
180 Kost Road
Mechanicsburg, PA 17055, USA

Fold and tape | Please do not staple | Fold and tape

IBM®



Diagnosis Reference
LC33-0517-0

Version 2.1

CICS/MVS

Program Number
5665-403

Printed in U.S.A.

LC33-0517-00