IBM

Customer
Information
Control
System
CICS/MVS

Licensed Program
Version 2.1.2

Program Number
5665-403

Customization Guide

SC33-0507-02

Third Edition (March 1991)

This edition applies to Version 2 Release 1 Modification 2 of the IBM licensed program Customer Information Control System/Multiple Virtual Storage (CICS/MVS), program number 5665-403, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

This book is based on the *Customization Guide* for CICS/MVS 2.1, SC33-0507-1. Changes from that edition are marked by vertical lines to the left of the changes.

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

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments 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.

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

Special notices

This book is intended to help you customize your CICS system. It contains information about user exits and user-replaceable programs. This book primarily documents Product-Sensitive Programming Interface and Associated Guidance Information provided by CICS.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

However, this book also documents General-Use Programming Interface and Associated Guidance Information and Diagnosis, Modification, and Tuning Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of CICS. General-Use programming interface information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

General-Use Programming Interface

General-Use Programming Interface and Associated Guidance Information

End of General-Use Programming Interface

This book also documents Diagnosis, Modification, and Tuning Information, which is provided to help you to customize CICS.

Warning: Do not use this Diagnosis, Modification, and Tuning Information as a programming interface.

Diagnosis, Modification, and Tuning Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Diagnosis, Modification, and Tuning Information

Diagnosis, Modification, and Tuning Information...

End of Diagnosis, Modification, and Tuning Information

Preface

What this book is about

This book provides the information needed to enhance and extend (or "customize") a CICS/MVS* 2.1.2 system. Customization includes generating management modules, coding service routines and exit programs, and using CICS commands to monitor and modify attributes of your system.

Who should read this book

This book is for those responsible for extending and enhancing a CICS system.

What you need to know to understand this book

To use the information in Part 1 of this book, you should understand the CICS installation process, which is described in the *CICS/MVS Installation Guide*. To use the information in Parts 2 through 6 of this book, you will need to be familiar with the architecture of CICS and the programming interface to CICS. Programming information can be found in the *CICS/MVS Application Programmer's Reference* manual.

Resource definition information can be found in the *CICS/MVS Resource Definition (Online)* manual and *CICS/MVS Resource Definition (Macro)* manual. For information on the purpose of specific CICS components, see the *CICS/MVS Diagnosis Reference* manual.

How to use this book

The parts and chapters of this book are self-contained. You should use an individual part or chapter as a guide when performing the task described in it.

Notes on terminology

This publication uses the following terms:

CICS/MVS	Customer Information Control System/Multiple Virtual Storage
IMS	IMS/VS and IMS/ESA*
VTAM*	ACF/VTAM* and the record interface of ACF/TCAM
TCAM	TCAM and the DCB interface of ACF/TCAM.

* IBM Trademark. For a list of trademarks see page iii.

Book structure

CICS/MVS 2.1.2 library

General

CICS Library Guide GC33-0356-04
Master Index SC33-0513-01
User's Handbook SX33-6061-01
Messages and Codes SC33-0514-02

Evaluation and planning

Brochure GC33-0503-00
CICS General Information GC33-0155-01
Facilities and Planning Guide SC33-0504-01
Release Guide GC33-0505-03
Data Tables General Information SC33-0684

Administration

Installation Guide SC33-0506-01
Customization Guide SC33-0507-02
Resource Definition (Online) SC33-0508-01
Resource Definition (Macro) SC33-0509-02
Operations Guide SC33-0510-01
CICS-Supplied Transactions SC33-0511-01

Special topics

Intercommunication Guide SC33-0519-02
Recovery and Restart Guide SC33-0520-01
Performance Guide SC33-0521-01
XRF Guide SC33-0522-02
CICS Communicating with CICS OS/2 SC33-0736-1
Data Tables Guide SC33-0632-01

Service

Problem Determination Guide SC33-0516-01
Diagnosis Handbook LX33-6062-01
Diagnosis Reference LY33-6077-00
Data Areas LY33-6078-00

Programming

CICS Application Programming Primer SC33-0674-00
Application Programmer's Reference SC33-0512-01

Version 1 books

CICS/VS Application Programmer's Reference Manual (Macro Level) (SC33-0079)

CICS/OS/VS IBM 3270 Data Stream Device Guide (SC33-0232)

CICS/OS/VS IBM 4700/3600/3630 Guide (SC33-0233)

CICS/OS/VS IBM 3650/3680 Guide (SC33-0234)

CICS/OS/VS IBM 3767/3770/6670 Guide (SC33-0235)

CICS/OS/VS IBM 3790/3730/8100 Guide (SC33-0236)

Related libraries

You may find the following books useful when you customize your CICS/MVS 2.1.2 system.

OS/VS TCAM Application Programmer's Guide, GC30-3036

OS/VS TCAM Installation and Migration Guide, GC30-3039

OS/VS TCAM System Programmer's Guide, GC30-2051

ACF/TCAM Installation and Migration Guide, SC30-3121

ACF/TCAM System Programmer's Guide, SC30-3117

ACF/TCAM Version 3 Application Programming, SC30-3233

ACF/VTAM Planning and Installation Reference, SC27-0584

ACF/VTAM Version 3 Programming, SC23-0115

ACF/VTAM Installation and Resource Definition, SC23-0111

OS/VS2 MVS JCL, GC28-0962

OS/VS2 System Programming Library: Debugging Handbooks, GBOF-3821

OS/VS2 System Programming Library: Debugging Handbook Volume 1 , GC28-1047

OS/VS2 System Programming Library: Debugging Handbook Volume 2 , GC28-1048

OS/VS2 System Programming Library: Debugging Handbook Volume 3 , GC28-1049.

OS/VS2 MVS Programming Library: Job Management, GC28-0627.

IBM ESA/370 Principles of Operation, SA22-7200

IMS/ESA Application Programming: DL/I Calls, SC26-4274

MVS/XA Introduction to Extended Recovery Facility (XRF), GC28-1135.

MVS/XA Debugging Handbook Volume 1, LC28-1164.

MVS/XA Debugging Handbook Volume 2, LC28-1165.

MVS/XA Debugging Handbook Volume 3, LC28-1166.

MVS/XA Debugging Handbook Volume 4, LC28-1167.

MVS/XA Debugging Handbook Volume 5, LC28-1168.

MVS/ESA Resource Measurement Facility (RMF), Version 4.1.1 - Monitor I & II Reference and User's Guide, LY28-1007

MVS/ESA SPL: Application Development Guide, GC28-1852

MVS/ESA SPL: Application Development Macro Reference, GC28-1857

MVS/ESA SPL: System Management Facilities (SMF), GC28-1819

RACF Macros and Interfaces, SC28-1345

System Programming Library: Resource Access Control Facility (RACF), SC28-1343.

Contents

Summary of changes

This edition is based on the *CICS/MVS Customization Guide* (SC33-0507-1), and incorporates updates and revisions as well as enhancements introduced by CICS/MVS 2.1.1 and CICS/MVS 2.1.2. These enhancements are described in the *CICS/MVS Release Guide*.

The opportunity has also been taken to correct errors and incorporate readers' comments.

All changes that are new in this edition, other than editorial changes, are marked by revision bars in the left margin, like this paragraph.

Changes to structure of book

As compared with SC33-0507-1, the following structural changes have been made.

"Chapter 4.10. User-replaceable conversion module for CICS/MVS-CICS OS/2 link" has been deleted. All material relating to links between CICS OS/2* and CICS host products is now in the *Communicating with CICS OS/2* manual.

Chapters 5.2 through 5.8 have been renumbered to 5.4 through 5.10 respectively.

The following new chapters have been introduced:

"Chapter 5.2. Exit to allow modification and redirection of CICS messages"

"Chapter 5.3. File control status exits"

"Chapter 5.11. Finding programs that use CICS macros"

* IBM Trademark. For a list of trademarks see page iii.

Questionnaire

CICS/MVS Version 2 Release 1 Modification 2 Customization Guide **Publication No. SC33-0507-02**

To help us produce books that meet your needs, please fill in this questionnaire. A reader's comment form is also included at the back of this book should you want to make more detailed comments. Whichever form you use, your comments will be sent to the author's department for review and appropriate action.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information 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. Which topics does the book handle well?

3. And which does it handle badly?

4. How could the book be improved?

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

6. What sort of work do you use CICS for?

7. How long have you been using CICS?_________years/months

8. Have you any other comments to make?

Thank you for your time and effort. No postage stamp necessary if mailed in USA. (If you are outside the USA, please mail this form to your local IBM office or representative who 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.) Be sure to print your name and address below if you would like a reply.

Name...Job Title ....

Company..Address..

....Zip...

Readers' Comments
SC33-0507-02

IBM®

Cut
Alo

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 6R1H
180 KOST ROAD
MECHANICSBURG PA 17055-0786

Fold and Tape | **Please do not staple** | Fold and Tape

Cut o
Along

Part 1. System generation

Introduction

Diagnosis, Modification, and Tuning Information

The CICS system is supplied on a distribution volume, which contains a pregenerated system library.

The following chapters describe the system generation macro (DFHSG). You can use this macro in the CICS system generation process to customize some parts of the pregenerated system provided by CICS. In "Chapter 1.2. DFHSG PROGRAM=xxx" on page 13, the modules of programs that have no options placed after them are the ones that cannot be generated in a modified form using DFHSG. You can request extra optional source materials to enable you to regenerate these modules without options. This process produces modules in the same form in which they appear in the pregenerated system. Certain other modules cannot be generated at all using DFHSG.

You should first install the pregenerated system, and only customize individual modules if your installation's requirements are not met. Some information about the pregenerated versions is included in the following chapters, but for fuller information about the pregenerated system, see the *CICS/MVS Installation Guide*.

The system generation process consists of three steps:

1. Coding a set of CICS system generation (DFHSG) macro instructions to specify the programs to be added or modified and to indicate how they should be tailored to meet your needs.
2. Assembling the macro instructions.
3. Executing the job stream that results from assembly of the macro instructions.

The process of assembling the macro instructions and executing the resultant job stream for CICS/MVS 2.1.2 is described in the *CICS/MVS Installation Guide*. The *CICS/MVS Resource Definition (Macro)* manual describes the system initialization table (SIT), which is also used to choose program options. You should also consult this book for more information about the different program versions available in the pregenerated system.

The next three chapters describe the DFHSG operands used to generate CICS system programs. The macro instructions are described in the following order:

- DFHSG TYPE=INITIAL
- DFHSG PROGRAM=xxx in alphabetic order of program name
- DFHSG TYPE=FINAL.

In each case, the TYPE or PROGRAM operand appears first. If the instruction has further operands, they are in alphabetic order. The macro format and syntax notation are described in "Syntax notation" on page 534.

The modules generated by each DFHSG command are not listed in Part 1 of this manual. For ease of reference there are two lists in Appendix A, "Program generation summary" on page 503. The first list is ordered alphabetically by the name of the module generated. The second list is ordered by the DFHSG PROGRAM= keyword. Part 1 contains comments on some of the modules generated by the individual DFHSG PROGRAM=xxx macros.

Examples of coding DFHSG macro instructions

For examples of how to code the DFHSG macro instructions, see the list of macro instructions used to define the pregenerated system, which are provided in member DFHSG04 of CICS212.MACLIB.

Chapter 1.1. DFHSG TYPE=INITIAL macro

```
DFHSG TYPE=INITIAL
      [,ASMBLR={IEV90|assembler-name}]
      [,CICSSVC={216|number}]
      [,DEBCHK={YES|NO}]
      [,{DL1|DLI}={NO|REMOTE|string}]
      [,EJECT={YES|n}]
      [,MTSLIB={YES|NO}]
      [,MOD=(module,suffix[,module,suffix]...)]
      [,OPSYS=MVS/XA]
      [,PREFIX=({CICS212|dprefix},[tprefix])]
      [,PRINT=([{LIST|NOLIST}]
              [,{XREF|NOXREF|SHORTXREF}]
              [,{DSECT|NODSECT|SOMEDSECT}]
              [,DSLIST])]
      [,PROCNMS=(DFHASMVS,DFHLNKVS,DFHSMPE|procedure-names)]
      [,SMPZONE=({GPREF|globalzone},{TPREF|zonename})]
      [,SRBSVC={215|number}]
      [,STAGE2={FORCE|SELECTIVE}]
      [,STARTER=YES]
      [,VTAM={YES|NO}]
```

Purpose

You should prefix each set of system generation macro instructions with a DFHSG TYPE=INITIAL macro instruction. You can reuse the procedures you develop from using this macro instruction for subsequent generations of the entire system or for parts of the system.

Modules generated

The following modules are generated in response to this macro instruction:

- DFHHPSVC — the service request block (SRB) type 6 supervisor call (SVC) for the high performance option (HPO)
- DFHCSVC — the bootstrap type 2 SVC
- DFHASV — the page fix/free SVC routine.

Note: Stage 2 jobs will always be produced for DFHHPSVC and DFHCSVC.

Operands

You should use the DFHJOB macro to specify job card information. The DFHJOB macro will be invoked by Stage 1 for each Stage 2 job produced. You should edit the DFHJOB macro and customize it to your own requirements. The *CICS/MVS Installation Guide* provides details of this operation. Use the DFHJOB macro to specify accounting information and a jobname prefix.

TYPE=INITIAL

Indicates that this is the initial macro instruction in a CICS system generation run.

ASMBLR={<u>IEV90</u>|assembler-name}

Code this with the name of the assembler to be used during Stage 2 of system generation and to produce the proper job control language (JCL). The default assembler for the MVS/XA* operating system is IEV90.

CICSSVC={<u>216</u>|number}

Code this to specify the type 2 SVC number to be used for the CICS bootstrap SVC that CICS will provide. This SVC is required if page-fixing is to be used; that is, if ANTICPG=YES or ANTICPG=number is coded in DFHPCT TYPE=ENTRY, if RES=FIX is coded in DFHPPT TYPE=ENTRY, or if FIX=YES is coded in DFHALT TYPE=ENTRY or DFHNLT TYPE=ENTRY, and for CICS monitoring facility. The SVC is also required for the multiregion operation (MRO) facility, the high performance option (HPO), the extended recovery facility (XRF), Resource Access Control Facility (RACF) support, and the device end program for the IBM* 7770 Audio Response Unit. The number may be in the range 200 through 255; the default is 216.

You must also code CICSSVC in DFHSIT. See the *CICS/MVS Resource Definition (Macro)* manual.

This operand controls the name given to the SVC routine that is generated by the DFHSG TYPE=INITIAL macro.

DEBCHK={YES|NO}

Applies only to 7770 devices. DEB checking is required and has a default of YES. DEBCHK=NO can only be coded if there are no 7770 devices on the MVS* system.

YES

The DEB validity check facility is supported.

NO

The DEB validity check facility is not supported.

{DL1|DLI}={<u>NO</u>|REMOTE|string}

Code this to specify whether the Data Language/I (DL/I) interface is to be included in this generation of CICS. The default is DLI=NO. You may code DLI=NO on any DFHSG invocation to produce non-DL/I versions of modules for the group specified.

This parameter is required if IMS is being used.

<u>NO</u>

DL/I support is not required.

REMOTE

DL/I support is required. All the databases that are to be accessed reside on remote CICS systems and are to be accessed through intercommunication support.

* IBM Trademark. For a list of trademarks see page iii.

DLI=YES must also be coded in DFHSIT or as a startup override when DLI=REMOTE is used. The BUFPL, DLTHRED, DMBPL, ENQPL, PISCHD, PSB, and PSBPL operands need not be coded in DFHSIT. However, an empty data management block directory list (DDIR) is required, together with a program specification block directory list (PDIR) that contains details of remote PSBs.

string
A string in the form n.n.n (where n is a single digit). The string indicates the level of IMS/VS or IMS/ESA for which CICS/MVS 2.1.2 support is to be included. CICS/MVS 2.1.2 supports IMS/VS Versions 1.3, 2.1, and 2.2, and IMS/ESA 3.1. The string values for these are 1.3.0, 2.1.0, 2.2.0, and 3.1.0 respectively. Information about the IMS release levels required for specific XRF and IMS functions is supplied in the *CICS/MVS Release Guide* and the *CICS/MVS XRF Guide*.

If you code DLI=string, DLI=NO may be coded in any DFHSG PROGRAM=group macro instruction to suppress the generation of DL/I-dependent modules from that group.

EJECT={YES|n}
Code this to specify the effect of page ejects in the assembly listings of the CICS modules. The default is EJECT=YES.

This operand can save paper by reducing the size of the CICS module listings depending on the value chosen for "n". This operand has no effect if you code PRINT=NOLIST.

YES
Normal page ejects will occur.

n A number from 2 to 99, controlling the number of spaces to be substituted for page ejects. A separator line preceded and followed by a "space x" statement (where x = n-2) will replace page ejects.

MTSLIB={YES|NO}
This operand specifies whether the CICS macro temporary storage (MTS) data set is included in the overrides for the SYSLIB concatenation in the stage 1 output of the system generation process

The MTS data set includes service that has been applied but not accepted. *Failure to include CICS212.MTS in the SYSLIB concatenation can cause assembly errors for IMS-related modules.*

YES
Overrides for the SYSLIB concatenation include the CICS MTS library.

Assuming the default prefix *CICS212* is in effect, the data sets in the SYSLIB concatenation are:

CICS212.MTS
CICS212.MACLIB
CICS212.SOURCE
IMSVS.OPTIONS (if DLI=*string* is coded)
IMSVS.GENLIB (if DLI=*string* is coded)
IMSVS.GENLIBA (if DLI=*string* is coded)

IMSVS.GENLIBB (if DLI=*string* is coded)
SYS1.MACLIB
SYS1.AMODGEN.

NO

Overrides punched for the SYSLIB concatenation do not include the CICS MTS library. Apart from CICS212.MTS, the data sets included in the SYSLIB concatenation are the same as those shown for MTSLIB=YES.

MOD=(module,suffix[,module,suffix]...)

Code this operand if the Stage 1 output produced by DFHSG will consist only of the jobs for those modules named in this operand. All other Stage 2 jobs will be suppressed. Stage 2 jobs for a module named in this operand will be suppressed unless the SUFFIX operand in the appropriate DFHSG PROGRAM=xxx macro instruction corresponds to the suffix parameter in the MOD operand. This allows fixes for authorized program analysis reports (APARs) to be applied to individual versions of the modules produced by DFHSG PROGRAM=xxx macro instructions.

Note: Only the TCP group of programs and DBP are suffixable.

module

The abbreviated name of a CICS module (for example, ZCY for DFHZCY). This name refers to the module generated by system generation macros (see Appendix A, "Program generation summary" on page 503) and does not refer to the PROGRAM operand of the DFHSG macro instruction.

The name specified must be that of an individual CICS module, and not that of a group. To produce output for a program group, all the module names in that group must be specified.

suffix

The optional suffix appended to the module. If this parameter is omitted, an unsuffixed version of the module will be searched for in the Stage 1. If ALL is coded, all Stage 1 versions will be dealt with. Only the TCP group of programs and DBP are suffixable, but when the suffix parameter is omitted, a comma must still be coded.

Example:

```
DFHSG TYPE=INITIAL,MOD=(ALP,,KCP,,SIA1,,PCP,,DBP,1$,RLR,A$)
DFHSG PROGRAM=KCP
DFHSG PROGRAM=CSO
DFHSG PROGRAM=PCP
DFHSG PROGRAM=DBP,SUFFIX=1$
DFHSG PROGRAM=BMS,BMSFUNC=STANDARD
DFHSG TYPE=FINAL
```

will produce Stage 1 jobs for DFHALP, DFHKCP, DFHSIA1, DFHPCP, DFHDBP1$, and DFHRLRA$ and will suppress Stage 1 jobs for:

- DFHKCSP, DFHSPP, DFHSPZ (that is, other KPP modules)
- All other CSO modules
- DFHRTY in DBP group
- All other BMS modules.

OPSYS=MVS/XA

The OPSYS operand identifies the environment in which CICS is to operate. MVS/XA is the required value for CICS/MVS 2.1.2, and will be assumed if you omit the operand. The default assembler for MVS/XA is IEV90. Note that the OPSYS option MVS/XA allows CICS/MVS 2.1.2 to run under either MVS/XA or MVS/ESA*.

PREFIX=({CICS212|dprefix},[tprefix])

Code this with the index names for the CICS system data sets. The JCL generated specifies these data sets as tprefix.LOADLIB, tprefix.LOADLIB1, tprefix.LOADLIB2, dprefix.MACLIB, dprefix.SOURCE, dprefix.SYSPUNCH, and dprefix.DLOADLIB, where 'dprefix' (distribution prefix) and 'tprefix' (target prefix) must conform to the data set naming conventions (maximum 35 characters).

If 'tprefix' is omitted, it will default to the value specified for 'dprefix', which in turn defaults to CICS212.

PRINT=([{LIST|NOLIST }][,{XREF|NOXREF|SHORTXREF}] [,{DSECT|NODSECT|SOMEDSECT}][,DSLIST])

Code this to specify the printing option for the assembly of the CICS modules during Stage 2 of system generation.

LIST
The total assembly listing is to be printed.

NOLIST
No assembly listing is produced. **NOLIST, if coded, overrides all options in the XREF and DSECT groups.**

XREF
The cross-reference list is to be printed.

NOXREF
No cross-reference list is to be printed.

SHORTXREF
The cross-reference list is to contain only symbols that are referenced.

DSECT
All CICS DSECTs are to be printed for each program.

NODSECT
None of the CICS DSECTs will be printed.

SOMEDSECT
The large DSECTs (CSA, TCA, TCTLE, and TCTTE) are not to be printed.

DSLIST
One listing will be printed of all the DSECTs. DSECTs are suppressed in each of the generated modules when NODSECT is in force. When DSLIST is coded, a job to assemble the DSECTs listing (DFHDSCTS) is generated from the DFHSG TYPE=INITIAL macro instruction.

* IBM Trademark. For a list of trademarks see page iii.

PROCNMS=(name1,name2,name3)
Code this to specify the names of CICS cataloged procedures:

name1
Assembles CICS programs and user-written assembler language programs. The default name is DFHASMVS.

name2
Link-edits CICS programs and application programs. The default name is DFHLNKVS.

name3
Executes the system modification program. The default name is DFHSMPE.

SMPZONE=({GPREF|globalzone},{TPREF|zonename})
Code this to provide information about the SMP/E database, known as the SMPCSI or CSI.

The **globalzone** is the prefix for the VSAM data set containing the SMP/E global zone (the top level of the CSI) that controls your CICS libraries. The globalzone identifier must not be more than 38 characters long.

This value will be specified on the GZONE parameter of the DFHSMPE procedure invoked in stage 2 of system generation. At that time it is suffixed with the characters '.CSI' to form the cluster name of the global zone data set. The default globalzone value is **GPREF**.

The **zonename** is the name of the SMP/E target zone that you want to update. The zonename value must not be more than 7 characters long, and the default value is **TPREF**.

This value will be specified on the ZNAME parameter of the DFHSMPE procedure invoked in stage 2 of system generation. At that time it is used to generate a SET BDY statement, which directs SMP/E to the target zone on which subsequent SMP/E processing is to be done. The target zone that you specify must correspond to the set of target libraries that the stage 2 jobs will update; that is, to the value of 'tprefix' on the PREFIX operand.

For an explanation of how CICS is installed and serviced under SMP/E see the *CICS/MVS Installation Guide.*

SRBSVC={215|number}
Code this with the SRB SVC number to be used for invoking the service request block (SRB) routine provided by CICS. This routine (DFHHPSVC) must be link-edited into your MVS nucleus. It is required to obtain access to the (SRB-dependent function) VTAM authorized-path HPO. The number coded must be in the range 200 to 255. You must also code SRBSVC and ZCP=HPO in DFHSIT. See the *CICS/MVS Resource Definition (Macro)* manual. If SRBSVC=number is coded, CICSSVC=number is also required.

STAGE2={FORCE|SELECTIVE}
Code this if DFHSG is to produce stage 2 jobs for programs requested. The option coded in this macro sets the defaults for the STAGE2 operands of other system generation macros. (Only DFHSG PROGRAM=DBP and DFHSG PROGRAM=TCP are affected by this operand.) The default for DFHSG TYPE=INITIAL is SELECTIVE.

The STAGE2 operand is not relevant when STARTER=YES is coded. If you code STARTER=YES, this implies STAGE2=FORCE. A DFHSG TYPE=INITIAL macro with the MOD operand should be used, therefore, when selectively generating pregenerated system modules.

FORCE
The stage 2 jobs are to be generated for any system generation programs requested. This must be coded if the IBM-supplied pregenerated system is not being used.

<u>SELECTIVE</u>
Stage 2 jobs may be selectively suppressed.

STAGE2=SELECTIVE causes DFHSG to suppress generation of the stage 2 job for any module supplied in the CICS pregenerated system. MNOTEs produced during the stage 1 assembly indicate which jobs have been suppressed and which jobs have been generated.

STARTER=YES
Code this if pregenerated system modules (with $ suffixes) are to be generated and various MNOTEs are to be suppressed. This operand must be used when service is to be applied to pregenerated system modules.

VTAM={<u>YES</u>|NO}
Code this to specify whether ACF/VTAM support is required. The default is YES.

<u>YES</u>
ACF/VTAM support is required.

NO
ACF/VTAM support is not required.

Chapter 1.2. DFHSG PROGRAM=xxx

BFP — built-in functions program

```
DFHSG PROGRAM=BFP
```

Purpose

DFHSG PROGRAM=BFP generates the built-in functions program, which provides the following facilities:

- Table search
- Verification of a data field — verify alphabetic or numeric
- Editing of a data field — removing unwanted characters
- Phonetic conversion
- Bit manipulation
- Input formatting
- Weighted retrieval function, which allows the user to search a specified group of records on a VSAM data set and to select only those records that satisfy specified criteria.

Notes:

1. The phonetic code conversion program (DFHPHN) is an offline subroutine which provides the facility to convert a 16-character name to a 4-byte phonetic code. See the "Built-in function" macro instruction DFHBIF TYPE=PHONETIC in the *CICS/VS Application Programmer's Reference Manual (Macro Level)* for the rules of the conversion.
2. The field-separator and field-name start characters are used for input formatting. The field-separator characters are specified by the FLDSEP operand of the DFHSIT macro instruction. The field-name start character is specified by the FLDSTRT operand of the DFHSIT macro instruction.

BMS — basic mapping support program

```
DFHSG PROGRAM=BMS
 [,BMSFUNC={MINIMUM|STANDARD|FULL}]
```

Purpose

The basic mapping support program, which can be generated by DFHSG PROGRAM=BMS, provides BMS functions. Code the BMSFUNC operand to select a level of BMS support, and from that you select the level of support you want in your system by using the BMS parameter on the system initialization table (SIT). See the *CICS/MVS Resource Definition (Macro)* manual. Note that you cannot select the FULL version on the SIT if you have selected the MINIMUM or STANDARD version in SYSGEN. If you select the full version of BMS, you will also require the temporary storage control program (TSP). The pregenerated version of BMS provides minimum, standard, and full versions, and you use the SIT to select your level of support in the same way as if you had used SYSGEN.

The modules are allocated as follows:

For the minimum version:

- Mapping control program (DFHMCP).

For the standard version (in addition):

- Page build program (DFHPBP)
- Non-3270 input mapping (DFHIIP)
- IBM 3270 Information Display System mapping (DFHM32)
- Data stream builder (DFHDSB)
- Route list resolution (DFHRLR)
- Terminal page program (DFHTPP)
- Fast path module (DFHMCX)
- LUI printer mapping (DFHMLI)
- Partition handling program (DFHPHP).

For the full version (in addition):

- Terminal page clean-up (DFHTPQ)
- Terminal page retrieval (DFHTPR)
- Terminal page scheduling (DFHTPS).

Operand

BMSFUNC={MINIMUM|STANDARD|FULL}
This indicates which version of BMS is to be generated. The default is FULL.

MINIMUM
Code this for the minimum version of BMS.

STANDARD
Code this for the standard version of BMS.

__FULL__
Code for the full version of BMS.

CSA — common system area

```
DFHSG PROGRAM=CSA
```

Purpose

The common system area can be generated by DFHSG PROGRAM=CSA.

In addition to generating the CSA, the execution of this macro instruction causes the assembly of terminal control's TCA, task control's TCA, and a write-to-operator (WTO) routine.

CSO — control system operational group

```
DFHSG PROGRAM=CSO
      [,CAA=appendage-suffix]
```

Purpose

The control system operational group can be generated by DFHSG PROGRAM=CSO.

Notes:

1. In non-BTAM systems, MSGIPK097 referring to the BTAM WAIT macro should be ignored in the Stage 2 assembly of DFHSTP.
2. The modules generated by this command that concern DL/I are only generated if DL/I is specified in the DFHSG TYPE=INITIAL command.
3. The 7770 read/write program is only generated if the CAA operand is coded.
4. The 7770 channel/abnormal end appendage program (IGG019zz where zz is the value coded for the CAA operand) is only generated if the CICSSVC and CAA operands are coded.
5. The 7770 device end program (DFHDEB70) is generated only if the CICSSVC and CAA operands are coded.
6. To use the message switching program (DFHMSP), basic mapping support must be generated with BMS=FULL on the SIT. In addition, the temporary storage control program is required.
7. Ensure that the assemblies for DFHEAI and DFHEAI0 are complete before link-editing DFHRCEX.

8. The DFHCRC, the interregion abnormal exit program, is generated in this group. DFHCRC includes the CICS SVC number, which must agree with that installed in the system (for details, refer to the *CICS/MVS Installation Guide*).

Operands

CAA=appendage-suffix

Code this with the 2-character alphanumeric suffix to be assigned to the 7770 channel end/abnormal end appendage routine provided by CICS when that routine is link-edited into CICS.LOADLIB by the Stage 2 job. The module is then known as IGG019xx, where xx is the suffix coded. The suffix coded must be in the range WA to Z9. This operand is required if the ACCMETH=BTAM and BTAMDEV=7770 operands are included in DFHSG PROGRAM=TCP, and if the APPENDG operand is included in DFHTCT TYPE=SDSCI. For information on adding appendages to the operating system, see *OS/VS2 System Program Library: Data Management*, GC26-3830.

Note: IGG019xx must be copied from CICS.LOADLIB into SYS1.LPALIB for MVS.

CSS — control system service group

```
DFHSG PROGRAM=CSS
```

Purpose

The control system service group can be generated by DFHSG PROGRAM=CSS.

CSU — control system utility group

```
DFHSG PROGRAM=CSU
```

Purpose

The control system utility group can be generated by DFHSG PROGRAM=CSU.

Support for all device types is generated for DFHDUP, DFHSTUP, and DFHTUP.

DBP — dynamic transaction backout program

```
DFHSG PROGRAM=DBP
      [,{DLI|DL1}=NO]
      [,STAGE2={SELECTIVE|FORCE}]
      [,SUFFIX=xx]
```

Purpose

DFHSG PROGRAM=DBP generates the dynamic transaction backout programs.

The function of the dynamic transaction backout program is to back out the effects of a single in-flight task that terminates abnormally, and to restore protected resources, that were altered by the task that failed, to the state they were in at the beginning of the logical unit of work (LUW). This feature operates while the rest of the CICS system is functioning normally, and not, as in the case of the transaction backout program, when emergency restart is invoked after CICS is unable to terminate normally.

There is one pregenerated version: DFHDBP1$, which has no DL/I support. DFHDBP2$, which does have DL/I support, is generated as part of the CICS system generation for full DL/I support.

Operands

{DLI|DL1}=NO
You can code DLI=NO to remove the DL/I support you specified in DFHSG TYPE=INITIAL. A dynamic transaction backout program without DL/I support is generated in this way.

STAGE2={SELECTIVE|FORCE}
May be used to override the specification of the default, set by the STAGE2 operand of DFHSG TYPE=INITIAL for producing the stage 2 job stream of this program.

SELECTIVE
Indicates that the stage 2 job stream for this program will be suppressed if that version of the program already exists on the pregenerated system.

FORCE
Forces generation of all stage 2 jobs for this program.

SUFFIX=xx
When you code the suffix, only DFHDBP receives it. SUFFIX=xx provides a 1-or 2-character suffix for the program being generated. If you generate your own version of DBP, do not use special characters or the reserved characters NO and DY. Do not use $ or # as the first suffix character.

DCP — dump control program

```
DFHSG PROGRAM=DCP
```

Purpose

The dump control program can be generated by DFHSG PROGRAM=DCP.

DIP — batch data interchange program

```
DFHSG PROGRAM=DIP
```

Purpose

The batch data interchange program, which can be generated by DFHSG PROGRAM=DIP, supports data communication between application programs running under CICS and logical units such as the IBM 6670, 3770, and 3790 Data Communication Systems.

The batch data interchange program also provides data management functions used with the 6670, 3790 and 3770 logical units.

The batch data interchange program must also be generated when a batch logical unit requires BMS features.

EIP — exec interface program

```
DFHSG PROGRAM=EIP
```

Purpose

The DFHSG PROGRAM=EIP macro generates an EXEC interface program, which supports the functions that can be accessed by the application programmer's command interface.

This is needed when the intercommunication facilities or the enhanced master terminal support (CEMT, CEST, and CEOT transactions) are being used.

Notes:

1. Generation of PL/I shared library support is described in the *CICS/MVS Operations Guide*.
2. You must assemble DFHEAI and DFHEAI0 before you link-edit the EIP modules given in the list under EXP below.

EXP — command (EXEC) language translator program

```
DFHSG PROGRAM=EXP
```

Purpose

DFHSG PROGRAM=EXP can generate a translator for the command interface to application programs written in PL/I, COBOL, or assembler. If your application programs use the command interface to CICS, you will also need the EXEC interface program. For further details, see the installation manuals for the appropriate compilers.

Notes:

1. DFHSG PROGRAM=HLL need only be generated if the macro interface to CICS is being used for PL/I and COBOL programs.
2. Translators for all languages are supplied in the pregenerated version of this program.
3. Generation of PL/I shared library support is described in the *CICS/MVS Operations Guide*.
4. DFHEAI and DFHEAI0 are generated from this program group even when support for the assembler HLPI is not requested. This is because EDF (generated from EIP) requires these assembler stubs at link-edit time.

   Ensure that the assemblies for DFHEAI and DFHEAI0 are complete before link-editing the following modules:

   - DFHRCEX (in CSO)
   - DFHRMSY (in KPP)
   - DFHMIR, DFHMXP (in ISC)
   - DFHBRCP, DFHECID, DFHECIP, DFHECSP, DFHEDAD, DFHEDAP, DFHEDFBR, DFHEDFD, DFHEMTA, DFHEMTD, DFHEMTP, DFHEOTP, DFHESTP (in EIP).

GAP — graphics attention program

```
DFHSG PROGRAM=GAP
```

Purpose

The graphics attention program can be provided by the DFHSG PROGRAM=GAP macro, which must be issued only if support for local IBM 2260 Display Stations is to be generated. This macro is not required under TCAM.

HLL — high-level language support group

```
DFHSG PROGRAM=HLL
```

Purpose

The high-level language support group, which can be generated by DFHSG PROGRAM=HLL, allows the COBOL or PL/I application programmer to use the macro interface to CICS. If application programs use only the command interface to CICS, the high-level language support group is not required. For details of the command interface to CICS, see DFHSG PROGRAM=EIP and DFHSG PROGRAM=EXP earlier in this part.

Notes:

1. The CICS preprocessor program (DFHPRPR) can be used for COBOL, or PL/I, or both.
2. Shared library transfer vector (PLISHRE) interfaces between PL/I optimizer code and its shared library modules.
3. The pregenerated version of this program provides full support for COBOL and PL/I.
4. Generation of PL/I shared library support is described in the *CICS/MVS Operations Guide*.

ICP — interval control program

```
DFHSG PROGRAM=ICP
```

Purpose

DFHSG PROGRAM=ICP generates the interval control program.

If interval control requests are used to store data for a future task, the temporary storage program must be available on the system.

ISC — intercommunication group

```
DFHSG PROGRAM=ISC
```

Purpose

The DFHSG PROGRAM=ISC macro instruction provides support for the CICS intercommunication facilities, where communication takes place between CICS systems or between CICS regions within a system.

In addition, DFHSG PROGRAM=TCP must be generated with ACCMETH=VTAM and VTAMDEV=LUTYPE6 when a connection, through ACF/VTAM, is required either in the same domain or cross-domain. DFHSG PROGRAM=TCP with ACCMETH=IRC must be coded when a region-remote connection, through the multiregion operation (MRO) facility, is required for CICS regions within the same processing unit.

The DFHSG PROGRAM=ISC macro is also required for CICS shared database.

In addition, DFHSG PROGRAM=TCP must contain ACCMETH=IRC.

Notes:

1. The following modules are for use with the DL/I shared database facility:
   - Batch transformer program (DFHXFQ)
   - Dependent region program (DFHDRP)
   - Batch region controller modules (DFHDRPA through DFHDRPG).
2. DFHSG macro instructions for PROGRAM=EIP and PROGRAM=EXP must precede that for PROGRAM=ISC, either as part of the same job or as an earlier job.

JCP — journal control program

```
DFHSG PROGRAM=JCP
```

Purpose

The journal control program can be generated by DFHSG PROGRAM=JCP.

Notes:

1. The pregenerated version of this program provides support for:

   Automatic journaling
   Dynamic transaction backout
   NOTE requests.

2. The SIT keyword DTB=AUX|MAIN controls the handling of spilled records when the dynamic buffer is overloaded. AUX implies that temporary storage auxiliary storage is to be used, MAIN that the spilled records are to be held in virtual storage. If you are using an XA machine, virtual storage is always used.

KCP — task control program

```
DFHSG PROGRAM=KCP
```

Purpose

The task control program can be generated by DFHSG PROGRAM=KCP.

KPP — keypoint program

```
DFHSG PROGRAM=KPP
```

Purpose

DFHSG PROGRAM=KPP generates programs associated with recovery.

Notes:

1. A DFHSG macro instruction for PROGRAM=EXP must precede the instruction for KPP, because DFHEAI and DFHEAI0 are required on the SYSPUNCH data set. You may generate EXP as part of the same job, or you can generate it earlier.
2. The pregenerated version provides full support for activity keypointing.

MTP — master terminal program

```
DFHSG PROGRAM=MTP
```

Purpose

The DFHSG PROGRAM=MTP generates the master terminal program, which is used by the master terminal (CSMT), supervisory terminal (CSST), and operator terminal (CSOT) transactions.

PCP — program control program

```
DFHSG PROGRAM=PCP
```

Purpose

DFHSG PROGRAM=PCP generates the program control program.

Note: This macro provides support for assembler, COBOL, and PL/I application programs, including American National Standards Institute COBOL V3 and V4. It also supports HLL trace. The pregenerated version also provides all this support.

PREGEN — starter system generation

```
DFHSG PROGRAM=PREGEN
```

Purpose

DFHSG PROGRAM=PREGEN is for IBM use, and is not intended for general use. You may require it when service is being applied to pregenerated system modules.

SCP — storage control program

```
DFHSG PROGRAM=SCP
```

Purpose

The DFHSG PROGRAM=SCP macro instruction generates the storage control program.

SRP — system recovery program

```
DFHSG PROGRAM=SRP
```

Purpose

The system recovery program, generated by DFHSG PROGRAM=SRP, is a generalized abnormal-termination handler. During CICS initialization, it issues the ESPIE and ESTAE macro instructions. During execution it is given control by the operating system if a program interruption or an operating system abend occurs.

Note: If SRT=NO is coded in the system initialization table or as an operator override, then SRP will take the required dumps, but will not provide any recovery action for program checks or for operating system abends.

TBP — transaction backout program

```
DFHSG PROGRAM=TBP
      [,{DLI|DL1}=NO]
```

Purpose

There are five transaction backout programs, generated by DFHSG PROGRAM=TBP. They are: DFHUSBP, DFHTSBP, DFHTCBP, DFHFCBP, and DFHDLBP. They are responsible for backing out changes made to CICS protected resources by transactions that were in-flight when the system was interrupted. This program must be generated if the keypoint program is included in your system.

The transaction backout programs are required components of emergency restart, which includes the collection of messages to allow message recovery. Further information about the backout programs (but not DFHTSBP) may be found in "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.

Operand

{DLI|DL1}=NO

You can code DLI=NO to remove the DL/I support you specified in DFHSG TYPE=INITIAL.

If you code DLI=NO, you will not generate DFHDLBP.

TCP — terminal control program

```
DFHSG PROGRAM=TCP
      ,ACCMETH=(method [,method],...)
      [,ANSWRBK=(identification[,identification],...)]
      [,AUTOTRN={NO|YES}]
      [,BSCODE=([EBCDIC][,ASCII])]
      [,BTAMDEV=(device[,device],...)]
      [,CHNASSY={NO|YES}]
      [,CONVTAB=([ABB][,ABC][,2741EU][,2741EM]
                  [,2741CU][,2741CM])]
      [,DEVICE=(device[,device],...)]
      [,EODI={E0|xx}]
      [,FEATURE=(feature[,feature],...)]
      [,INITRL=YES]
      [,LOCKF=YES]
      [,LOGREC={NO|YES}]
      [,PIPELN={NO|YES}]
      [,PUNSOL={NO|YES}]
      [,STAGE2={SELECTIVE|FORCE}]
      [,SUFFIX=xx]
      [,TBLFIX={NO|YES}]
      [,TCM3270=YES]
      [,TWXOFF=xx]
      [,TWXON=xx]
      [,UCTRAN={NO|([EBCDIC][,ASCII])}]
      [,VTAMDEV=(device[,device],...)]
      [,WRAPLST={NO|YES}]
```

Purpose

The DFHSG PROGRAM=TCP macro instruction generates the terminal control program.

Modules generated

The programs generated are as follows:

For all access methods:

- Console activity control (DFHZCNA)
- Console application request (DFHZCNR)
- 3270 print function support (DFHP3270)
- Terminal control (DFHZCP and DFHZCX). *

For ACCMETH=VTAM:

- Good morning message program (DFHGMM)
- 3270 print function support modules:
  - DFHRKB
  - DFHCPY
  - DFHPRK
  - DFHEXI.
- RPL executor in SRB mode (DFHZHPRX)
- Resend program (DFHZRSP)
- Response logging program (DFHZRLG)
- Terminal control program modules:
  - DFHZCA
  - DFHZCB *
  - DFHZCC *
  - DFHZCW *
  - DFHZCY
  - DFHZCZ. *
- Node abnormal condition program (DFHZNAC)
- Node error program (DFHZNEP). *

For ACCMETH values other than VTAM:

- Terminal control program (DFHTCP). *

Notes:

1. Stage 2 jobs will always be produced for the modules marked "*".
2. If SRBSVC=number is coded in DFHSG TYPE=INITIAL, VTAM authorized path is used in DFHZCP to give improved performance characteristics.
3. DFHTCP, DFHZCP, DFHZCB, DFHZCX, and DFHZCZ will receive a suffix when the SUFFIX=xx operand is coded.
4. Four pregenerated versions of TCP are provided:
   - A$, covering SAM (all devices), BTAM (local 3270)
   - B$, covering SAM (all devices), BTAM (local 3270, remote 3270, IBM 3275 Display Station dial-up)

- E$, covering SAM (all devices), VTAM (3270, 3790, IBM 3600 Finance Communication System), but with no support for LUTYPE6 protocols
- S$, covering all access methods and all devices, and including support for LUTYPE6.1 and LUTYPE6.2 protocols.

5. In the pregenerated system, 7770 devices are not supported. You must use SYSGEN options if you use this device.

Operands

ACCMETH=(method[,method],...)

Code this to identify the access method(s) to be used in the terminal environment. One or more of the following keyword parameters must be coded:

Method	Required
TCAM	–
BTAM	BTAMDEV
BSAM	DEVICE
SAM	DEVICE
BGAM	–
VTAM	VTAMDEV
IRC	–

SAM and BSAM are functionally synonymous in CICS, and can be used interchangeably. Only unblocked data sets can be used with SAM or BSAM. BGAM provides 2260 support.

Notes:

1. Do not code ACCMETH=VTAM if VTAM=NO was coded in DFHSG TYPE=INITIAL.
2. ACCMETH=VTAM and VTAMDEV=LUTYPE6 must be coded for a connection, via ACF/VTAM, that is either in the same domain or cross-domain.
3. ACCMETH=IRC generates control code in the group of DFHZCP modules for the DL/I shared database interregion control module and for region-remote connections when the multiregion operation (MRO) facility is being used. If any other ACCMETH value is coded, this control code is automatically generated, and ACCMETH=IRC can be omitted.

ANSWRBK=(identification[,identification],...)

Code this with the type of terminal identification. This must be used if FEATURE=AUTOANSW is coded. The parameters of this operand are not mutually exclusive. This operand is applicable only when ACCMETH=BTAM is coded.

AUTOMATIC

Automatic terminal identification is to be sent by the terminal. This option is only valid for BTAMDEV=TWX.

EXIDVER

BTAM expanded identification verification is to be used to identify those terminals that transmit unique identification sequences. ANSWRBK=EXIDVER may be coded for all BTAM BSC dial devices (except for the IBM 2780 Data Transmission Terminal) that require the expanded ID verification feature.

TERMINAL
The operator will supply the identification for switched lines.

7770TERM
The operator will supply the terminal identification.

7770NULL
No terminal identification is to be sent by either the terminal or by the operator; instead, the terminal control program will connect the line to the next available terminal in the terminal pool. The default is ANSWRBK=7770TERM, providing BTAMDEV=7770 has also been coded.

Note: The ANSWRBK operand must include all keyword parameters for which the corresponding parameter is to be included in DFHTCT TYPE=LINE.

AUTOTRN={NO|YES}
Code this if the optional automatic transaction initiation feature is to be included in CICS. The default is AUTOTRN=NO.

NO
Automatic transaction initiation is not required.

YES
Automatic transaction initiation is required.

BSCODE=([EBCDIC][,ASCII])
Code this for the types of binary synchronous communication code to be supported when ACCMETH=BTAM is coded. The default is BSCODE=(EBCDIC,ASCII).

EBCDIC
Extended Binary Coded Decimal Interchange Code.

ASCII
American Standard Code for Information Interchange.

BTAMDEV=(device[,device],...)
Code this to identify the IBM BTAM device types. This operand must be coded if ACCMETH=BTAM is coded. The applicable keyword parameters are:

1050	IBM 1050 Data Communication System.
1050D	1050 Data Communication System (dial-up).
1053	IBM 1053 Printer on a Local/Remote IBM 2848 Display Control Unit.
2260	2260 Display Station (Remote).
2265	IBM 2265 Display Station.
2740	IBM 2740 Communication Terminal Model 1.
2740D	2740 Communication Terminal Model 1 (dial-up).

2740-2	2740 Communication Terminal Model 2 (2740 must also be specified).
2741C	IBM 2741 Communication Terminal with correspondence code.
2741E	2741 Communication Terminal with PTTC/EBCD code.
2741DC	2741 Communication Terminal with correspondence code (dial-up).
2741DE	2741 Communication Terminal with PTTC/EBCD code (dial-up).
2770	IBM 2770 Data Communication System.
2770D	2770 Data Communication System (dial-up).
2780	2780 Data Transmission Terminal.
2780D	2780 Data Transmission Terminal (dial-up).
2980/1	IBM 2980 General Banking Terminal System Model 1.
2980/2	2980 General Banking Terminal System Model 2.
2980/4	2980 General Banking Terminal System Model 4.
3275	3275 Display Station (remote).
3275D	3275 Display Station (dial-up).
L3270	Local support for IBM 3276 Control Unit Display Station, IBM 3277, 3278, and 3279 Display Stations and IBM 3284, 3286, 3287, 3288, and 3289 Printers.
R3270	Remote support for 3276, 3277, 3278, and 3279 Display Stations and 3284, 3286, 3287, 3288, and 3289 Printers.
3600	3600 Finance Communication System.
3660	IBM 3660 Supermarket Scanning System.
3735D	IBM 3735 Programmable Buffered Terminal (dial-up).
3740	IBM 3740 Data Entry System.
3740D	3740 Data Entry System (dial-up).
3780	IBM 3780 Data Communication Terminal.
3780D	3780 Data Communication Terminal (dial-up).
7770	7770 Audio Response Unit Model 3.
SYS/3	IBM System/3 Models 6 and 10.
SYS/3D	System/3 Models 6 and 10 (dial-up).
SYS/7	IBM System/7.
SYS/7D	System/7 (dial-up).
S/370	IBM System/370.*
S/370D	System/370 (dial-up).

* IBM Trademark. For a list of trademarks see page iii.

S/7BSCA	System/7 with Binary Synchronous Communications Adapter.
S/7BSCAD	System/7 with Binary Synchronous Communications Adapter (dial-up).
TLX	Teletypewriter (WTC only). The Autocall feature is not supported by CICS. (This feature is for World Trade users only.)
TWX	IBM CPT Teletypwriter Exchange System (Model 33/35).

Note: In the codes, the prefixes L (for local) and R (for remote) are used with the four-digit IBM product number of each device in the 3270 family.

Individual device type parameters are provided for the BTAMDEV operand so that system generation input is self-documenting. If the parameter length for this operand exceeds the assembler limit of 255 characters for the particular system being generated, synonymous parameters can be omitted. Specifying any one of the parameters from a group produces code for all devices in the group. These groups are:

SYS/3, S/370, BISYNC, S/7BSCA, SYS/3D, S/370D, S/7BSCAD,
3660, 2260, 2265.

CHNASSY={<u>NO</u>|YES}
Code this if a complete SNA chain of logically grouped records is to be read before any of the input data is presented to the application program. This operand is only to be used when ACCMETH=VTAM is coded. The default is CHNASSY=NO.

<u>NO</u>
Chain assembly support is not to be generated in the ZCP group of modules.

YES
Chain assembly support is to be generated in the ZCP group of modules.

CONVTAB=([ABB][,ABC][,2741EU] [,2741EM][,2741CU][,2741CM])
Code this with the type of conversion to be performed on the data received from the 7770 Audio Response Unit or the 2741 terminal.

- If BTAMDEV=7770, CONVTAB=ABB and/or ABC applies. The default is CONVTAB=(ABB,ABC).
- If BTAMDEV=2741E and/or 2741DE, CONVTAB=2741EU and/or 2741EM applies. The default is CONVTAB=2741EU.
- If BTAMDEV=2741C and/or 2741DC,CONVTAB=2741CU and/or 2741CM applies. The default is CONVTAB=2741CU.

ABB
Conversion from ABB transmission code.

ABC
Conversion from ABC transmission code.

2741EU
Data received from a 2741 EBCDIC terminal will be translated to uppercase.

2741EM
Data received from a 2741 EBCDIC terminal will be translated to text mode.

2741CU
Data received from a 2741 correspondence terminal will be translated to uppercase.

2741CM
Data received from a 2741 correspondence terminal will be translated to text mode.

Note: The 2741 Autocall feature is not supported by CICS.

DEVICE=(device[,device],...)
Code this to identify the direct access or sequential devices that are to be used in the terminal environment. This operand must be used if ACCMETH=SAM or ACCMETH=BSAM is coded. The applicable parameters are: CRLP (card reader, line printer), DASD, and TAPE.

EODI={E0|xx}
Code this with the end-of-data indicator for sequential input. The characters xx represent two hexadecimal characters in the range 01 to FF. The default is EODI=E0, which is equivalent to the 0-2-8 punch formerly used as an end-of-data indicator.

FEATURE=(feature[,feature],...)
Code this with the special features present in the terminal environment. The applicable keyword parameters are:

AUTOANSW
Automatic answer feature. This enables a control unit to respond automatically to a call received over a switched line. When BTAM is used, this feature is required for the 3275 and for all dialed devices.

AUTOPOLL
Automatic polling feature. When BTAM is used, this feature is required for multipoint BSC terminals.

BUFFRECV
Buffered receive feature for 2740 Model 2.

PSEUDOBIN
Pseudobinary transmission code for System/7.

RDATT
2741 Read Attention feature.

TRANSPARENCY
Character transparency for the 2770, 2780, 3600 BSC, S/3, S/370, and S/7BSCA.

WRBRK
2741 Write Break feature.

INITRL=YES
Code this if all reads from other than an application program are with the keyboard lock option. The FEATURE=KBRDLOCK operand must be included in DFHTCT TYPE=LINE to have the keyboard lock feature operative for that line. This operand applies only to the 2260 "family" of devices.

LOCKF=YES
Code this to specify that the optional keyboard lock feature, supporting 2848 models 21 and 22, is to be included in CICS. The FEATURE=KBRDLOCK operand must be included in DFHTCT TYPE=LINE to make the keyboard lock feature operative for that line. This operand applies only to 2260 devices.

LOGREC={NO|YES}
Code this if deblocking input records (so that the application program can read each logical record) is to take place. The default is LOGREC=NO. This operand only applies when ACCMETH=VTAM is coded.

NO
Logical record presentation support is not to be generated in the ZCP group of modules.

YES
Logical record presentation support is to be generated in the ZCP group of modules.

PIPELN={NO|YES}
Code this if 3600 or IBM 3650 Retail Store System pipeline session support is required. The default is NO.

NO
3600 or 3650 pipeline session support is not required.

YES
3600 or 3650 pipeline session support is required. PIPELN=YES is required for the IBM 3606 Financial Services Terminal, the IBM 3608 Printing Financial Services Terminal, and the IBM 3653 Point of Sale Terminal pipeline sessions. SESTYPE=PIPELINE must also be indicated in DFHTCT TYPE=TERMINAL.

PUNSOL={NO|YES}
Code this with YES to protect the 3270 logical unit from receiving unsolicited input. The default is PUNSOL=NO.

In normal operation, the 3270 terminal operator is expected to wait until the keyboard is unlocked by a reply from the application program before attempting to enter further input. Use of the reset key to allow further input before the application program replies is not regarded as normal use of the terminal. Specifying PUNSOL=YES will protect application programs from receiving such unsolicited input (which may cause a synchronization problem between the operator and the application program). CICS can check whether such unsolicited data has been received and can discard it, without giving any indication that it was received. CICS will not check for unsolicited data for an LUTYPE2 device (which is handled as a

3270-compatibility-mode logical unit) because the compatibility-mode controller function protects CICS from unsolicited input.

<u>**NO**</u>
Protection is not required.

YES
Protection is required.

STAGE2={SELECTIVE|FORCE}
May be used to override the specification of the default, set by the STAGE2 operand of DFHSG TYPE=INITIAL for producing the stage 2 job stream of this program.

SELECTIVE
Indicates that the stage 2 job stream for this program will be suppressed if that version of the program already exists on the pregenerated system.

FORCE
Forces generation of all stage 2 jobs for this program.

SUFFIX=xx
Provides a 1-or 2-character suffix for the program being generated. Do not use special characters or the reserved suffixes NO and DY. Do not use $ or character.

TBLFIX={<u>NO</u>|YES}
Code this if the 2980 translate tables are to be generated. The default is TBLFIX=NO.

<u>**NO**</u>
Skeleton translate tables will be generated and used to build the translate tables dynamically each time input or output is converted. This is to conserve storage.

YES
A set of preassembled tables will be used for better performance.

TCM3270=YES
Code this if TCAM support includes the 3270 Information Display System.

TWXOFF=xx
Code this to generate instructions to handle the transmit-off character, which is specified by the CHAREC=(XOFF,xx) parameter when the EP/3705 is generated for a TWX terminal.

Note: The TWXOFF and TWXON operands (see below) need only be coded when BTAMDEV=TWX is coded.

TWXON=xx
Code this to generate instructions to handle the transmit-on character, which is specified by the CHAREC=(XON,xx) parameter when the EP/3705 is generated.

Note: All generated EP/3705 lines used by CICS must have the same value coded for the CHAREC parameter.

UCTRAN={NO|([EBCDIC][,ASCII])}

Code this to generate instructions to translate lowercase data to uppercase in 3270, 3767, and 3770 SDLC input data streams. The default is UCTRAN=NO.

NO

Uppercase translation is not required.

EBCDIC

EBCDIC support is to be generated, when FEATURE=UCTRAN is coded in DFHTCT TYPE=TERMINAL for:

- VTAM 3270s
- SDLC 3767s and 3770s
- Non-VTAM 3270s when BSCODE=EBCDIC and/or CONVTAB=EBCDIC is coded in DFHTCT TYPE=LINE.

BSCODE and CONVTAB do not apply for 3270 or LUTYPE2 logical units, so UCTRAN=EBCDIC will generate translation support for all 3270s.

ASCII

Code this if support is to be generated for BTAM 3270s. For BSC 3270s, translation is available by means of NCP translation tables in the 3704/3705. There is no support for ASCII-encoded data received from 3270 compatibility mode logical units.

Uppercase translation for the 3270, 3767, or 3770 SDLC devices is only performed on input data streams received from those devices for which FEATURE=UCTRAN is coded in DFHTCT TYPE=TERMINAL, except to satisfy DFHTC TYPE=TEXT or terminal control ASIS requests. Translation is not performed on data copied from a display to a printer.

VTAMDEV=(device[,device],...)

Code this to identify the logical units. This must be coded if ACCMETH=VTAM is coded. The applicable keyword parameters are:

3600	3600 Finance Communication System.
3614	IBM 3614 Consumer Transaction Facility.
3650	3650 Retail Store System.
3790	3790 Communication System.
3270	3270 Information Display System. (Does not include support for 3270s running as LUTYPE2, LUTYPE3, or SCSPRT logical units.)
BCHLU	Batch logical unit support.
3770	3770 Data Communication System (batch logical unit).
3770B	3770 Data Communication System (batch logical unit).
INTLU	Interactive Logical Unit (flip-flop mode).
3767	IBM 3767 Communication Terminal operating as INTLU.
3767C	3767 Communication Terminal operating as an interactive logical unit in contention mode.

3767I	3767 Communication Terminal operating as INTLU.
3770C	3770 Data Communication System (terminals 3771, 3773, 3774, 3775 only) operating as an interactive logical unit in contention mode.
3770I	3770 Data Communication System operating as INTLU.
LUTYPE2	SNA type 2 logical unit (3270-compatible logical unit).
LUTYPE3	SNA type 3 logical unit (3270 printer logical unit).
LUTYPE4	SNA type 4 logical unit.
LUTYPE6	Session Type 6 logical unit for ISC support.
SCSPRT	SCS printer logical unit (for example, 3287, 3289).

- INTLU generates support for 3767 and 3770 (terminals 3771, 3773, 3774, and 3775 only) interactive logical units in flip-flop mode, and for VTAMDEV=SCSPRT. 3767, 3767I, and 3770I may also be specified.
- BCHLU (or 3770 or 3770B) also generates support for the 3770 batch data interchange, LUTYPE4, and 3770 full function logical units.
- VTAMDEV=3790 generates support for LUTYPE2, LUTYPE3, and SCSPRT logical units.
- VTAMDEV=INTLU or VTAMDEV=3767C must be coded when TWX and TLX devices are to run as logical units under VTAM through the Network Terminal Option (NTO).
- VTAMDEV=LUTYPE6 generates support for both the type 6.1 and type 6.2 logical units.

WRAPLST={NO|YES}

Code this if the optional wrap list feature is to be included in CICS. Note that this option applies only to remote terminals. The list to be constructed is a wraparound polling list for a nonswitched line. The polling list is to be constructed in the terminal control table. This operand is for BTAM only. The default is WRAPLST=NO.

NO

A wrap list will not be used for polling.

YES

A wrap list will be used for polling.

TDP — transient data control program

```
DFHSG PROGRAM=TDP
```

Purpose

The transient data control program can be generated by DFHSG PROGRAM=TDP.

Notes:

1. Only **VSAM** intrapartition transient data is supported.
2. The pregenerated version provides a full function module, including extrapartition data set input and output, intrapartition queues, automatic transaction initiation, and recovery.

TRP — trace control program

```
DFHSG PROGRAM=TRP
```

Purpose

The trace control program, generated by DFHSG PROGRAM=TRP, is used for program maintenance and performance tuning. Used in conjunction with the trace utility program, this feature provides for easy use of CICS trace facilities.

Notes:

1. The pregenerated version provides support for writing trace table entries on to the auxiliary trace file.
2. A user program that interprets trace data online must execute in 31-bit addressing mode to address the trace table residing above the 16MB (megabytes) line.

TSP — temporary storage control program

```
DFHSG PROGRAM=TSP
```

Purpose

DFHSG PROGRAM=TSP may be used to generate the temporary storage control program.

Note: The pregenerated version provides full support for all functions.

Chapter 1.3. DFHSG TYPE=FINAL

```
DFHSG TYPE=FINAL
```

Purpose

Stage 1 is terminated in response to the DFHSG TYPE=FINAL macro instruction. This macro instruction must be the last statement of the system generation input stream preceding the assembler END statement. The assembler END statement does not require an operand.

Operand

TYPE=FINAL
Indicates the end of CICS Stage 1 system generation.

End of Diagnosis, Modification, and Tuning Information

Part 2. Writing recovery and restart routines

This part of the book provides information about the programming interfaces available to a user wishing to extend the CICS recovery and restart facilities. More information about recovery and restart is provided in the *CICS/MVS Recovery and Restart Guide*.

Table 1 summarizes the information in this part of the book.

Table 1 (Page 1 of 2).

Error Situation	CICS Function	Modifications Available to You	Chapter
Program check	Interception and transaction abend	See "Tranaction abend" below	
System abend	Interception, system recovery table, standard SRT routine	User-written SRT recovery code	"Chapter 2.1. Writing a system recovery table recovery routine or program" on page 43
Transaction abend	Program level abend exit invocation	User-written program level abend exit code	"Chapter 2.2. Program level abend exit" on page 47
	Program error program (DFHPEP)	User-written program error program	"Chapter 2.3. Writing a program error program (DFHPEP)" on page 51
	Dynamic transaction backout	User-written DTB exit code	"Chapter 2.4. Writing dynamic transaction backout exits" on page 53
	Transaction restart (DFHRTY)	User-written DFHRTY	"Chapter 2.5. Writing a transaction restart program (DFHRTY)" on page 57

Table 1 (Page 2 of 2).

Error Situation	CICS Function	Modifications Available to You	Chapter
System failure	Emergency restart with transaction backout	User-written exits for the transaction backout programs	"Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59
		User-written activity keypoints	"Chapter 2.7. Writing a user activity keypoint program" on page 67
Terminal error (BTAM/TCAM)	Sample terminal error program	User-written terminal error program	"Chapter 2.8. User-written utility to scan for unit of work ids" on page 69
Logical unit error (VTAM)	Sample node error program	User-written note error program	"Chapter 2.9. The terminal error program" on page 71
Region failure in an XRF environment	Sample overseer program	User-cutomized or -written overseer program	109

Chapter 2.1. Writing a system recovery table recovery routine or program

The CICS system recovery program (DFHSRP) receives control as a result of a program check or an operating-system abend. It can:

- Recover from the error and avoid shutdown by terminating the affected task
- Pass control to a user exit
- Take action to isolate the error, and make restart easier.

The program has two functions:

Program check recovery
If the program check was caused by an error in a CICS application-program task, CICS terminates the task with the code ASRA.

User exits that can be invoked are discussed in "Chapter 2.2. Program level abend exit" on page 47 through "Chapter 2.5. Writing a transaction restart program (DFHRTY)" on page 57.

If the program check is more serious, CICS is terminated.

Abend recovery
The action CICS takes for each abend depends on entries in the system recovery table (DFHSRT) and the routines or programs provided for processing the abends.

Note: Some functions, such as some RACF calls and some file control calls made to VSAM, are issued under an operating system subtask which handles its own abend recovery. System recovery table (SRT) processing is not involved in such a recovery.

Default system recovery table

The default system recovery table (SRT) provides:

A list of abend codes that CICS handles
This list is shown with the DFHSRT macro in the *CICS/MVS Resourc Definition (Macro)* manual.

An exit routine for these abends (DFHSRTRR)
The default routine tries to isolate the problem to a single CICS task and abnormally terminates that task with CICS abend code ASRB. If it cannot, CICS is terminated.

The default SRT is adequate for most users. However, you can add extra abend codes to be handled by the default routine or by your own routine or program. You extend the list of abend codes in the SRT by coding an additional entry.

For example:

```
DFHSRT TYPE=USER,ABCODE=999,ROUTINE=DFHSRTRR
```

This adds the user code 999 to the codes that the default routine handles.

Creating a recovery routine or program

The routine or program invoked as a result of an abend can be in one of two formats:

- An assembler-language routine coded inline with the SRT after TYPE=FINAL, or link-edited with the SRT.
- An assembler-language program with an entry in the CSD file or in the processing program table (PPT).

Notes:

1. The recovery routine is driven only if CICS is able to continue. If a recursive abend occurs (DFH0612), or if a system task is executing when an abend occurs (DFH0613), then the recovery routine is not driven.
2. The recommended method of coding is to use a routine rather than a separately assembled program, as any error within that program could cause CICS to terminate abnormally.

Adding entries to the system recovery table

To relate a routine or program to an abend, use a DFHSRT macro to add an entry to the SRT.

For a **routine coded inline or link-edited with the SRT**, use the ROUTINE operand of the DFHSRT macro.

For example:

```
DFHSRT TYPE=USER,ABCODE=997,ROUTINE=ABND997
```

This causes the routine ABND997 to be invoked if an abend 997 occurs.

The routine is entered via a BALR R14,R15 instruction.

For a **separately assembled program**, use the PROGRAM operand of the DFHSRT macro. Write the program to the CICS macro level interface in assembler language, and make an entry for it in the PPT, or in the CSD file. The program is entered via a DFHPC TYPE=LINK macro.

Coding considerations (ROUTINE and PROGRAM)

- DSECT generation

  For a separately assembled routine or program, you will need to provide COPY statements for the appropriate CICS control blocks, such as DFHCSADS and DFHTCADS.

  If you are assembling your routine inline with the DFHSRT macro, you must not have COPY statements for DFHCSADS, DFHTCADS, DFHFCTDS, DFHFWADS, or DFHDCTDS, because these are defined in the SRT itself. If you do include COPY statements for one or more of these DSECTs, assembler errors will result.

- Register equates

  For inline assembly with the SRT, you must not redefine registers SRTRRBAR, FWACBAR, FCTDSBAR, DCTCBAR, or WORKREG. If you do, you will receive assembler errors.

- Register save area

  If you are coding your routine inline with, or link-editing it with the SRT, you will probably need to provide a save area to save the caller's registers.

  For example:

```
ABND997     DS    0H
            USING *,15
            STM   14,11,SAVEAREA      ENTRY
            .
            .
            LM    14,11,SAVEAREA      EXIT
            BR    14
SAVEAREA    DS    14F
```

- Input parameters

  For both routines and programs:

  - Register 12 points to the TCA of the transaction that was in control when the abend occurred.
  - Register 13 points to the CSA.
  - Register 14 contains the return address.
  - Register 15 contains the entry point.
  - Register 1 points to a **copy** of the system diagnostic work area (SDWA).

    The SDWA is passed by MVS to ESTAE exit routines and contains information about the abend condition. For a description of the SDWA, see the relevant *MVS/XA Debugging Handbook*.

    To allow a program to work in more than one operating system environment, the program logic should test the field SDWACTL2 in the SDWA addressed by register 1. If the PSW in the field SDWACTL2 is 0, then MVS was unable to acquire an SDWA, and only the abend code in SDWAABCC is valid.

  - Operating system abend code.

    In all situations, the operating system abend code is stored in the abnormally terminating task's TCA at TCAATAC and can be interrogated by the recovery logic.

    Its format is 00xxxyyy, where xxx is the OS/VS system abend code and yyy is the hexadecimal representation of the user abend code.

    For example:

```
00B37000 is an OS/VS B37 abend
000001F5 is a user 501 (=X'1F5') abend
```

- Program logic

  You should make sure that the code that is included in your user program does not cause another abend condition, because this will force CICS to terminate abnormally.

  The flag byte, TCAPCARO, can be tested, and can be modified to pass return information to the calling routine.

  The following characters can be set in TCAPCARO:

"C"	Cancel any "program level abend exits" that are associated with this task. Allow the task to be abended "ASRB". Keep CICS up and running.
"P"	Do not cancel any "program level abend exits," but allow them to proceed. Allow the task to be abended "ASRB". Keep CICS up and running.
"any other value"	Allows CICS to be terminated abnormally.

- Returning control.

  To exit from a routine, you must restore the registers from your save area before returning via a BR R14 instruction.

  To exit from a program, you must exit using a DFHPC TYPE=RETURN macro instruction.

- Addressing mode

  The routine or program must be assembled with AMODE 24.

Chapter 2.2. Program level abend exit

General-Use Programming Interface

An exit routine established by a DFHPC TYPE=SETXIT macro instruction, or by an EXEC CICS HANDLE ABEND command will be executed if a program abend is requested while the task is at the level at which the SETXIT was issued, or at a lower level. If the task continues to abend, the program error program (DFHPEP) will be entered, if previously defined, after return from the highest level.

If an abend occurs while a transaction is being processed by the task-related user exit interface module DFHERM (for example, EXEC DLI commands and DB2* transactions), CICS takes a transaction dump even if an application-level HANDLE ABEND is in effect. If you want to suppress the dump, you can use the global user exit XPCABND (see page 313).

A transaction abend can occur because of:

- A user request by, for example:

  ```
  EXEC CICS ABEND ABCODE(...)
  ```

  or

  ```
  DFHPC TYPE=ABEND,ABCODE=...
  ```

- A CICS request as a result of an invalid user request, for example, an invalid FREEMAIN request, which gives error code "ASCF".
- A program check, in which case DFHSRP is driven, and the task is abended with code "ASRA". For full details, see the *CICS/MVS Problem Determination Guide*.
- An operating system abend, in which case DFHSRP is driven, and the task is abended with code "ASRB" (unless "C" was placed in TCAPCARO as described on page 46). For full details, see the *CICS/MVS Problem Determination Guide*.

The CICS processing described below is the same regardless of where the abend request originated. The program level abend exit or exits are executed, followed by DFHPEP. However, it should be pointed out that as each succeeding exit is entered, the logic is further away from the cause of the abend, and the available information and corrective action possible are reduced.

Program level abend exits will not be driven if a task is abnormally terminated with any of the following abend codes: ACMF, AKCP, ALFA, ASPE, ASPL, ASP1, ASP2, ASP3, ASP5, ASP6, ASP7, ASP8, or ASP9.

* IBM Trademark. For a list of trademarks see page iii.

Creating a program abend exit

The DFHPC TYPE=SETXIT macro instruction and the HANDLE ABEND command allow the application programmer to specify the name of a program or a routine to be given control when a task ends abnormally. Exit programs can be coded in any supported language, but exit routines must be coded in the same language as the program of which they are a part. On entry to the abend routine, the addressing mode is that in effect when the exit was invoked.

For information on the transaction abend codes for abnormal terminations that are initiated by CICS, their meanings, and your responses, see the *CICS/MVS Messages and Codes* manual.

Upon entry to an exit program, no addressability can be assumed other than that normally assumed for any application program coded in that language. If the exit logic is in the form of a routine (DFHPC TYPE=SETXIT,ROUTINE=...), the amount of addressability varies depending on the source language (for the macro interface) as follows:

- Assembler

Reg 12	TCA address
13	CSA address
14	Entry address for routine
15-11	Varies depending on cause and location of abend.

- COBOL

Reg 12	PGT address
13	TGT address
14	Entry address for routine
15-11	Contents at time of last CICS service request.

For a routine, the register values in the command interface (HANDLE ABEND LABEL (...)) are:

- Assembler

Reg 15	Abend label
0-14	Contents at the time of the last CICS service request, or at the time the last "call" was invoked.

- COBOL

  Control returns to the HANDLE ABEND command with the registers restored; a COBOL GO TO is then executed.

Other information that is available to the exit routine or program includes:

- The current abnormal completion code at TCACRABC (macro level interface only).
- The original abnormal completion code at TCAORABC (macro level interface only).
- Any user-defined information that is placed in the TWA.
- If the abnormal completion code is ASRA (that is, as a result of a program check), the PSW at the time of program interrupt is stored in field TCAPCPSW. (Macro level interface only.)

There are three means of terminating processing in an exit routine or program:

- DFHPC TYPE=RETURN or the RETURN command indicate that the task is to continue running with control passed to the program on the next higher logical level. If no such program exists, the task is terminated normally.
- DFHPC TYPE=ABEND and the ABEND command indicate that the task is to be abnormally terminated with control passed either to an exit specified for a program on a higher logical level or, if there is not one, to the abnormal condition program (DFHACP) for abnormal termination processing.
- A branch to retry an operation. When you are using this method of retrying an operation, and you wish to reenter the original exit routine or program if a second failure occurs, the exit routine or program should issue either the DFHPC TYPE=RESETXIT macro, or the HANDLE ABEND RESET command before branching. This is because CICS will have disabled the routine or program to prevent it reentering the exit. It is your responsibility to establish registers and code for the use of the exit logic.

  Note: If an abend occurs during the invocation of a CICS service, you should be aware that issuing a further request for **the same service** may cause unpredictable results, because the reinitialization of pointers and work areas, and the freeing of storage areas in the exit routine, may not have been completed.

End of General-Use Programming Interface

Chapter 2.3. Writing a program error program (DFHPEP)

The distributed version of the program error program (DFHPEP) contains code to establish a base register, to establish addressability to the system portion of the task control area (TCA), and to return control to DFHACP through a DFHPC TYPE=RETURN operation. DFHACP will not allow transactions beginning with 'C' to be disabled; you should not, therefore, attempt to disable CICS-supplied transactions.

Note that DFHPEP cannot influence the taking of a transaction dump.

You can modify the source of DFHPEP to include your own logic if you want. The DFHPEP module is a dummy module. If you want to customize DFHPEP, you have to code all the source yourself. To help you, we provide a listing of DFHPEP in Figure 1 on page 52. When you have written your program error program, assemble it, and use it to replace the supplied dummy program.

See the guidance on the preparation of application programs in the *CICS/MVS Operations Guide* for the job control statements necessary to assemble and link-edit these components.

Information available to DFHPEP includes:

- The current abend code at TCACRABC.
- The original abend code at TCAORABC.
- The program status word (PSW) at the time of program interrupt at TCAPCPSW (for abend code ASRA only).
- The program control table (PCT) entry address at TCATCPC.
- Any other data placed in the TWA by the application program or SETXIT routines.
- Register 1 points to a list of addresses:
  - First address is that of the 4-byte abend code
  - Second address is that of the PCT entry
  - Third address is that of the return value for PCT disabling (TCAPECOM).

  If the return value is X'01', the transaction is disabled (provided it does not begin with 'C').

If the PCT entry is to be disabled, a hexadecimal 01 should be placed in field TCAPECOM at the system portion of the TCA. For example:

```
MVI  TCAPECOM,TCAPEDIS      SHOW PCT TO BE DISABLED
```

Note: TCAPEDIS has been equated to X'01' in the TCA dummy section.

```
*
* REGISTER DEFINITION
*
PCTCBAR  EQU        8                  PCT BASE REGISTER
TCASBAR  EQU        9                  TCA SYSTEM AREA REGISTER
PEPBAR   EQU        10                 PEP BASE REGISTER
         DFHEJECT

*
* DUMMY SECTIONS
*
         DFHPRINT   DSCT=START
         COPY       DFHCSADS
         DFHTCA     CICSYST=YES
         COPY       DFHTERID
         COPY       DFHPCTDS
         DFHEJECT
         DFHPRINT   DSCT=END

*
* PROGRAM ERROR PROGRAM
*
DFHPEP   CSECT                         PROGRAM ERROR PROGRAM CSECT
         DFHVM      PEP                GENERATE HEADING CONSTANT
         ENTRY      DFHPEPNA           ESTABLISH ENTRY POINT
DFHPEPNA DS         0H                 ENTRY POINT
         BALR       PEPBAR,0           ESTABLISH ADDRESSABILITY
         USING      *,PEPBAR           ...AND BASE REGISTER
         L          TCASBAR,TCASYAA    LOAD TCA SYSTEM AREA ADDRESS
         USING      DFHSYTCA,TCASBAR
*
*   Insert your own code here
*
         DFHPC      TYPE=RETURN        ISSUE CICS RETURN MACRO
         LTORG      *
DFHPEPEA DS         0H                 MODULE END ADDRESS
         DFHEND     DFHPEPNA           ASSEMBLY END
```

Figure 1. Source code of program error program (DFHPEP)

Chapter 2.4. Writing dynamic transaction backout exits

The dynamic transaction backout program (DFHDBP) has four user exits that you can code if the default action does not suit your requirements. The method available for making use of user exits is described in "Chapter 5.1. Global user exits" on page 289. If an exit is not used, the default action corresponds to a return code of 0.

The four exits are:

1. **XDBINIT** This exit is given control on entry to DFHDBP. Valid return codes are:

   0 to continue dynamic transaction backout
   4 to suppress DL/I backout
   8 to suppress all backout.

   Note: For return codes 4 and 8, any databases updated by DL/I will be closed by backout failure processing. For details, see the *CICS/MVS Operations Guide*.

2. **XDBIN** This exit is given control when each log record (other than one from DL/I) is obtained. Register 3 points to the record read from the dynamic log. This record is mapped by DFHDBRDS DSECT. Valid return codes are:

   0 to continue processing the record
   4 to ignore the record (not applicable to the record corresponding to the input message).

3. **XDBFERR** This exit is given control when an error condition has been returned from the file control program during the backout processing or if an error has been detected by DFHDBP itself.

   Register 3 points to the record read from the dynamic log. The record should be referenced using DFHDBRDS DSECT. Valid return codes are:

   0 to accept error and continue
   4 to ignore error and continue
   8 to reapply the FWA version of the record.

   The byte DBRERRCD in the log record is set for different types of error as follows:

   DBFEGU
   Means that an error response has been returned from the file control program (FCP) while servicing a GET-UPDATE-request. DFHDBP has attempted to retrieve the existing copy of the record prior to backing it out. The file control CHECK macro in combination with the type of record pointed to by DBRREG ("read-for-update" or "write-add") can be used in the exit to determine the specific problem.

DBFELE
Means that the file work area (FWA) acquired from the FCP is not large enough to receive the before-copy data picked up from the dynamic log to perform the backout. The symbolic register FWACBAR points to the FWA on entry to the exit. The file control CHECK macro is not applicable to this error.

DBFEPU
Means that an error response has been returned from the FCP while servicing a PUT-UPDATE-request. DFHDBP has attempted to replace the existing copy of the record on the file with the "before-copy" pointed to by DBRREG. The file control CHECK macro can be issued in the exit to determine which error occurred.

DBFEPN
Means that an error response has been returned from FCP while servicing a PUT-NEW-request. DFHDBP has attempted to add the "before-copy" of a deleted VSAM KSDS record. The file control CHECK macro can be issued in the exit to determine the specific error.

DBFEWA
If the record read from the dynamic log is a WRITE-ADD, and the file accessed is a VSAM ESDS or a BDAM data set.

Note: (This condition also applies to an AIX path defined over a VSAM ESDS base.)

The dynamic transaction backout program (DFHDBP) reads the record from the file using a GET-UPDATE, but recognizes that no delete function exists for BDAM and VSAM entry-sequenced data sets (including a VSAM ESDS accessed by an AIX path). You are given the opportunity to "mark" the existing record on the file as deleted according to application-dependent logic. The FWA version of the record should be marked. If you want the FWA version to be reapplied, a return code of 8 should be specified.

Register 6 points to the FWA containing the existing record on the file. The file control CHECK macro is not applicable to the error.

DBFEVD
Means that an error response was returned from the FCP while servicing a VSAM-DELETE request. DFHDBP has attempted to delete a new record added to a VSAM key-sequenced data set. The file control CHECK macro can be issued in the exit to determine the specific error.

4. **XDBDERR** When the DL/I backout routine detects an error, its error message is routed to CSMT and this exit is then given control. Register 3 points at the corresponding dynamic log record. The information in the TCA fields TCADLII and TCADLIPA is also available. Valid return codes are:

   0 to suppress further DL/I backout
   4 as for return code 0.

Note: To preserve database integrity, any databases updated by DL/I will be closed by backout failure processing. For guidance see the *CICS/MVS Operations Guide.*

User-written dynamic transaction backout exits must be quasi-reentrant.

Recoverable resources may be modified in user exits but the following should be noted:

- Your exit code must not contain EXEC CICS ABEND requests, because these will cause CICS to terminate.
- Changes to recoverable transient data and temporary storage should be avoided in the XDBINIT exit because they will be backed out immediately.
- File control GET for updates should be properly released, either implicitly or explicitly, or else backout may be locked out.
- The current DL/I program specification block (PSB) should be left scheduled; it should not be terminated.

Register usage

The exit program should save and restore all registers it modifies, using the save area addressed by register 13.

Chapter 2.5. Writing a transaction restart program (DFHRTY)

The conditions under which CICS automatically restarts a transaction after dynamic transaction backout are described in the *CICS/MVS Recovery and Restart Guide*. If you wish to modify the conditions under which a transaction is restarted, you can edit the transaction restart program (DFHRTY); this program provides a framework for the user's assembler code.

See the guidance on the preparation of application programs in the *CICS/MVS Operations Guide* for the job control statements necessary to assemble and link-edit these components.

The distributed version of the program DFHRTY contains code to:

- Establish a base register
- Establish addressability to the system portion of the TCA
- Send a message to CSMT if restart is about to be attempted
- Return control to DFHDBP through a program control RETURN operation.

Information available to DFHRTY includes:

- Byte TCAZLUWT (status of the LUW) contains flags:

  TCAZRRD (read since last syncpoint)
  TCAZRWRT (write done since last syncpoint)
  TCAIOSK (syncpoint taken).

  Any of these flags set would normally prevent transaction restart.

- Byte TCADBRTS contains flags:

  TCADBTRD (task has previously been restarted).

  TCADBTRP (restart to proceed — can be set by user logic in DFHRTY after default setting has been performed by DFHDBP).

- TCAORABC (original abend code).
- TCACRABC (current abend code).

  These two values can be different if, for example, a task was restarted after program isolation deadlock, and encountered a program check after restart.

- Byte TCADBRTC contains a count of previous restarts on this task. User code may require you to test the count of restarts to ensure that recursive abends are not caused by restarting tasks that DFHDBP would not normally allow to proceed.

Dynamic transaction backout will suppress restart (when the abend code is other than that for program isolation or a syncpoint, or if terminal traffic after initial input has occurred) unless user-written exit code (DFHRTY) tells it to proceed by setting TCADBTRP. Otherwise, when transaction restart is used, all messages from dynamic transaction backout will be suppressed, and the task will be restarted from the beginning, with the following information available to it:

- The initial input TIOA (if any)
- The contents of the TCTUA and the command-level communications area, as at the start of the task
- The TCADBTRD flag (via ASSIGN RESTART command).

Chapter 2.6. User-written exits for resource backout or recovery at emergency restart

General description

User exits written for use during the backout of resources at recovery are global user exits. For more general information about global user exits, see "Chapter 5.1. Global user exits" on page 289.

At recovery, it is necessary to back out updates that were not committed when the system failed. There are six programs that perform this backout, and they run in parallel under their own CICS tasks. (The transient data and temporary storage backout programs are not included in this chapter because user exits are not involved.) This chapter describes exits in the following programs:

DFHFCBP
File control backout

DFHUSBP
User recovery, to back out user system log entries

DFHTCBP
Message and ISC state recovery

DFHDLBP
DL/I backout.

There are four exits that apply at backout. The initialization/termination exit is driven twice by all four programs. The input exit is not driven by DFHDLBP. The open error and file error exits apply only to DFHFCBP. See Figure 2 on page 60.

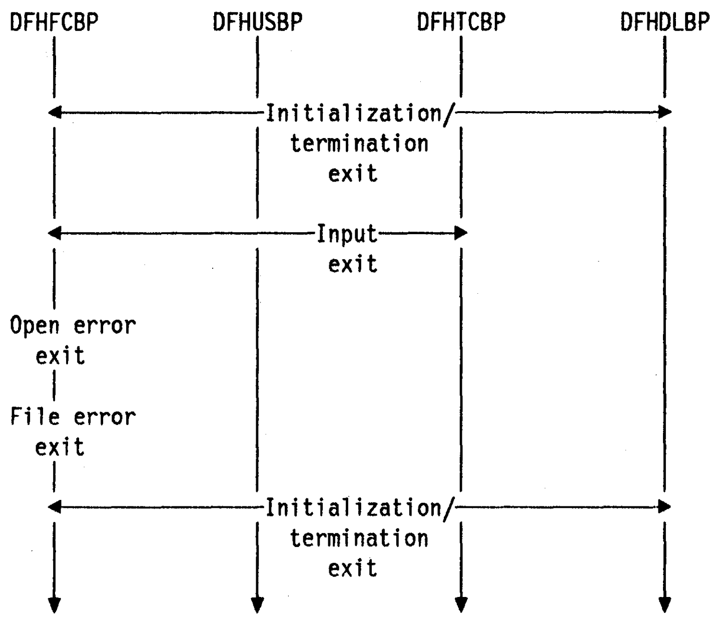


Figure 2. User exits for backout at recovery

The figure shows the four programs running in parallel. The initialization/termination exit is a single exit, which can be invoked from four different programs. The different invocations for the initialization/termination exit are distinguished by the value passed in register 2, as described below. Note that the same exit serves at initialization and termination. Whether the exit has been called at initialization or termination is also identified by the contents of register 2.

The input exit is a single exit invoked by three programs. CICS212.MACLIB provides two copybooks that you should include in your exit program:

DFHJCRDS containing a DSECT that describes log records, *and*

DFHFMIDS containing useful equates.

To determine which program has invoked the exit, see the log record addressed by register 10. The JCRSTRID field (defined in DFHJCRDS) is a 2-byte field. You can check the JCRSTRID+1 byte against the following fields defined by equates in DFHFMIDS:

MODIDFC	log record written by file control
MODIDTC	log record written by CICS terminal control
MODIDUSR	log record written by the user.

DFHFCBP will process file control records, DFHTCBP will process terminal control records, and DFHUSBP will process user log records.

The open error and file error exits can only be invoked by file control backout.

Before the description of the exits, there are some other points to keep in mind. You have access to all other CICS services, except terminal control services, during exit execution. But the use of temporary storage, transient data, file control, or DL/I is not recommended, because these resources may also be in a state of recovery and therefore "not open for business". Access to these services will therefore at best cause serialization of the recovery tasks and at worst cause a deadlock. In addition:

- An exit must not release, or cause to be released, any file control area (pointed to by Register 7) as a result of DFHFCBP processing.
- No exit should reset either the "absent" or "no action" indicators set by CICS in the backout tables.
- Only the initialization/termination exit can set the "no action" indicators in the file, message, or DL/I backout table entries.
- An exit must not attempt to make any file control requests to a VSAM data set with a string number of 1, unless 'no action' has been specified for that data set during the user initialization exit.

The initialization/termination exit, XRCINIT

The initialization/termination exit is invoked once at the beginning and once at the end of each of the backout programs. Its interface is:

- For initialization of user recovery:

  Register 2 contains X'00'.

  Register 7 addresses the transaction backout table (TBO), if any. Otherwise, register 7 = 0. The TBO is described by the COPY book, DFHTBODS.

- For termination of user recovery:

  Register 2 contains X'80'.

  Register 7 addresses the TBO, if any. Otherwise, register 7 = 0.

- For initialization of file recovery:

  Register 2 contains X'01'.

  Register 7 addresses the file backout table (FBO), if any. Otherwise, register 7 = 0.

  The FBO is described by the COPY book, DFHFBODS. The entries in the FBO have been verified against the loaded file control table and marked as "absent" and "no action" if unmatched.

  The exit may scan the FBO and mark additional files for "no action".

  Before giving control to the exit, DFHFCBP has listed the "absent" file IDs to the console operator.

- For termination of file recovery:

  Register 2 contains X'81'.

  Register 7 addresses the FBO, if any. Otherwise, register 7 = 0.

- For initialization of DL/I recovery:

  Register 2 contains X'02'.

  Register 7 addresses the DL/I backout table (DBO), if any. Otherwise, register 7 = 0.

  The DBO is described by the COPY book, DFHDBODS. The entries in the DBO have been verified against the loaded DL/I DMB and PSB directories and marked as "absent" and "no action" if unmatched.

  Before giving control to the exit, DFHDLBP has listed the missing or unschedulable PSB and DMB names to the console operator.

- For termination of DL/I recovery:

  Register 2 contains X'82'.

  Register 7 addresses the DBO, if any. Otherwise, register 7 = 0.

- For initialization of message recovery:

  Register 2 contains X'03'.

  Register 7 addresses the message backout table (MBO), if any. Otherwise, register 7 = 0.

  The MBO is described by the COPY book, DFHMBODS. The entries in the MBO have been verified against the loaded terminal control table and marked as "absent" and "no action" if unmatched.

  The exit may scan the MBO and mark additional entries for "no action".

- For termination of message recovery:

  Register 2 contains X'83'.

  Register 7 addresses the MBO, if any. Otherwise, register 7 = 0.

The input exit, XRCINPT

This exit applies to DFHFCBP, DFHUSBP, and DFHTCBP. The interface is as follows:

- Register 10 addresses the current log record, which is described by the DSECT DFHJCRDS.
- Register 7 addresses the FBO table entry for a file control record, and the MBO table entry for a terminal control record. Otherwise, register 7 is undefined.
- To determine which program has invoked the exit, see the log record addressed by register 10. In the log record, field JCRSTRID is a 2-byte field, and JCRSTRID+1 will contain one of the following equates (defined in DFHFMIDS):

MODIDFC	log record written by file control
MODIDTC	log record written by CICS terminal control
MODIDUSR	log record written by the user.

  DFHFCBP will process file control records, DFHTCBP will process terminal control records, and DFHUSBP will process user log records.

- On return, register 15 should contain a return code of 0 or 4.

The input exit is given control each time a record (other than a DL/I record) has been read from the restart data set. At that time, register 10 points to the record that should be referenced by using DSECT DFHJCRDS. The type of record can be determined by testing field JCRSTRID with the symbolic codes provided by "DSECT" DFHFMIDS.

If you want the default action upon return from the input exit, the return should have a return code of 0. If you want no action, a return code of 4 should be used, in which case the record area will be freed immediately and a new record will be read. The default actions are:

User journaled records	No action.
Automatically journaled records	No action, unless the record is also a logged record (see below).
Logged records applying to files or terminals flagged for "no action"	No action.
Logged "read-updates"	Reapply "before-copy" of the record to the file.
Logged "write-add"	The user's file error exit (see below) is given control after the file record has been read for update for BDAM and VSAM ESDS files. For VSAM KSDS files, the default action is to delete the record.
Logged temporary storage "PUT(Q)-REPLACE"	Reapply the "before-copy" of the record to temporary storage.
Logged terminal messages	Save the records in the temporary storage "resend slot" and/or "message cache", as appropriate.

The open error exit, XRCOPER

This exit is for program DFHFCBP only, to assist in file control backout. The interface is as follows:

- Register 7 addresses the FBO entry corresponding to the entry.
- Register 2 is undefined.

The exit is given control if an error occurs while opening a file control data set. A message has been written to CSMT and to the console operator with a "GO" or "CANCEL" option. The exit is only given control if the operator selects "GO". Upon return from the exit, DFHFCBP marks the file backout table entry "no action".

The file error exit, XRCFCER

This exit applies only to DFHFCBP. The interface is as follows:

- Register 2, as defined below
- Register 7 addresses the FBO entry
- Register 9 addresses the file work area (FWA), if applicable
- Register 10 addresses a copy of the log record
- Register 11 addresses the FCT entry, where indicated.

The file error exit is given control when an error condition has been returned from the file control program during the backout processing, or if an error has been detected by DFHFCBP itself.

Register 10 points to the record read from the restart data set, and should be referenced using DSECT DFHJCRDS. Register 7 points to the corresponding DFHFBO entry. Except as indicated below, the file error exit has no processing options, and the return code is ignored. Register 2 is primed for different types of errors, with the following symbolic values, the actual values of which are defined in DFHFBODS:

TBFEGU
If an error response is returned from the file control program while servicing a GET-UPDATE-request. DFHFCBP has attempted to retrieve the existing copy of the record before backing it out. The file control CHECK macro in combination with the type of record pointed to by register 10 ("before-copy" of a read-for-update record, or "new-copy" of a "write-add" to be deleted) can be used in the exit to determine the specific problem.

TBFELE
If the FWA acquired from the FCP is not big enough to receive the before-copy data from the restart data set to perform the backout. Register 9 points to the FWA on entry to the exit. The file control CHECK macro is not applicable to this error.

TBFEPU
If an error response is returned from the FCP while servicing a PUT-UPDATE-request. DFHFCBP has attempted to replace the existing copy of the record on the file with the "before-copy" pointed to by register 10. The file control CHECK macro can be issued in the exit to determine the specific error.

TBFEPN
If an error response is returned from the FCP while servicing a PUT-NEW-request. DFHFCBP has attempted to add the "before-copy" of a deleted VSAM KSDS record. The file control CHECK macro can be issued in the exit to determine the specific error.

TBFEWA
| If the record read from the restart data set is a WRITE-ADD, and the file
| accessed is a VSAM ESDS or a BDAM data set.

| **Note:** (This condition also applies to an AIX path defined over a VSAM ESDS
| base.)

The file control backout program (DFHFCBP) reads the record from the file using a GET-UPDATE, but recognizes that no delete function exists for BDAM and VSAM entry-sequenced data sets (including a VSAM ESDS accessed by an AIX path). You are given the opportunity to "mark" the existing record on the file as deleted according to application-dependent logic. You should mark the FWA-version of the record. If you want the FWA version to be reapplied, register 15 should contain a return code of 0. If you do not want this, but would rather bypass the operation, use a return code of 4.

Register 11 points to the file control table (FCT).

Register 9 points to the FWA containing the existing record on the file. The file control CHECK macro is not applicable to the error.

TBFEVD
If an error response is returned from the FCP while it is servicing a VSAM-DELETE request. The backout program has attempted to delete a new record added to a VSAM-key-sequenced data set. The file control CHECK macro can be issued in the exit to determine the specific error.

Chapter 2.7. Writing a user activity keypoint program

An activity keypoint is taken at the start of each system log volume or data set, and periodically after that. It records on the system log the information necessary:

- To restore recoverable resources during emergency restart
- To determine which tasks were in-flight at the time of the system failure.

Activity keypointing is done by attaching the CICS activity keypoint transaction (CSKP) at a predefined frequency. You can define this frequency at system initialization time with the AKPFREQ override or operand of DFHSIT. You can also alter this frequency during execution with the CEMT SET AKP command.

The frequency of the activity keypoint and the amount of logging performed by in-flight transactions determine the amount of log data to be processed at restart time, and thus the duration of the recovery process.

You can include your own keypoint records in the activity keypoint sequence. You do this by providing a user program DFHUAKP. This program should be used to record a limited amount of selected user data (that is, tables to be restored following an emergency restart). It should be written to avoid suspension of the keypoint task (that is, program and work areas should be resident). This program should issue only CICS journal control functions. Note that the first use of an activity keypoint should not rely on the results of any program in the PLT. In order to perform efficiently, the journal control requests should be asynchronous (that is, WRITE without WAIT) and with STARTIO=NO. This method will force synchronization by writing a synchronous end of keypoint record upon return from the user program. To make these records accessible to the DFHUSBP exits during emergency restart, you should assign your own identification to them. You can do this by means of the JTYPEID operand, with the high-order bit set on.

Note: Do not code DFHUAKP in either VS COBOL II or VS PL/I, because it can be invoked before these languages have been initialized.

Chapter 2.8. User-written utility to scan for unit of work ids

If a system failure in an interconnected system occurs during the syncpointing process, for a certain interval neither system knows if the other has committed the updates and whether it should commit its own. This period of time is called the **indoubt window**.

A unit of work (UOW) is the period between two syncpoints; each UOW is identified by a UOW identifier. This UOWID is written to the system log by each task when the task makes its first change to a recoverable resource. It is also included in any messages generated during an indoubt window failure.

You can therefore write a log-scanning utility to read the system log records for the UOW in the affected CICS region, to determine what action is needed to synchronize the databases.

Format of UOWID

The system log contains a UOWID record for each unit of work, denoting its start. This record has the following format:

System header	System prefix	Journalled data

System header: The record can be recognized as of type UOWID if the JCRSTRID has the value FIDLEUOW||MODIDSPP.

System prefix: The field JCSPTASK contains the task number associated with the current UOW.

Journaled data: This field contains the UOWID related to the current task in the format defined by TCAUOWDS.

System log processing algorithm

The UOWID is displayed in the CICS messages DFH2101/2/3/4, using the ISMUOWID format (see DFHIMSDS in the Data Areas). You should convert this UOWID to the TCAUOWDS format (see DFHTCA in the Data Areas) using the following algorithm.

```
TCAUOWL = ISMUWLEN            total length
          -( L'ISMUWC1         - length of fixed part ISM format
             + L'ISMUWTKN
             + L'ISMUWC2
             + L'ISMUWSEQ )
          +( L'TCAUCLK         + length of fixed part TCA format
             + L'TCAUNUM )

TCAULUNL = ISMUWLEN            total length
           -( L'ISMUWC1         - length of fixed part ISM format
              + L'ISMUWTKN
              + L'ISMUWC2
              + L'ISMUWSEQ )

TCAULUN = ISMUWLUN            Logical unit name - maximum 17 chars

TCAUCLK = HexCharToBinary(ISMUWTKN) Hex characters to packed

TCAUNUM = CharToBinary(ISMUWSEQ)    Dec characters to binary
```

Figure 3. System log processing algorithm

Note: Take great care — TCAULUN is a variable length data element that affects the **addresses** of TCAUCLK and TCAUNUM.

You should then process the system log in chronological order, looking for each UOWID record that contains a UOWID that matches the converted UOWID.

Make a note of the task number (JCSPTASK) in the system prefix. You need to locate all of the records that follow with the same task number. This search terminates on a Physical/Logical End of task record, identified by a JCRSTRID field of FIDPETK||MODIDSPP (X'F359') or FIDLETK||MODIDSPP (X'F259').

These records form the total recovered journaled data; they then need further processing to effect a recovery for a particular UOWID.

Chapter 2.9. The terminal error program

This chapter contains information on the CICS terminal error program (DFHTEP) that handles error conditions for devices that operate in a non-VTAM environment. The CICS-supplied sample terminal error program and the user-written versions of this program are discussed, as well as topics related to error conditions for specific device types.

CICS terminal error-handling is based on the assumption that most users want to modify certain CICS operations in response to various terminal errors. Because CICS cannot anticipate all possible courses of action, the error-handling facilities have been designed to allow maximum freedom for users to create unique solutions for errors that occur within a terminal network.

The following CICS components are involved in the detection BTAM and/or TCAM terminals are used:

- Terminal error program (DFHTEP)
- Terminal control program (DFHTCP)
- Terminal abnormal condition program (DFHTACP).

The corresponding CICS components for logical units are discussed in "Chapter 2.10. The node error program" on page 109.

Note: Node error programs, not terminal error programs, must be used for VTAM-supported devices.

When an abnormal condition occurs

When an abnormal condition associated with a particular terminal or line occurs, the terminal control program puts the terminal out of service and passes control to the terminal abnormal condition program (DFHTACP) that, in turn, passes control to a version of the terminal error program (DFHTEP, either CICS-supplied or user-written), so that it can take the appropriate action.

Terminal control program

When the terminal from which the error was detected has been put out of service, the terminal control program creates a terminal abnormal condition line entry (TACLE), which is chained off the real entry, the terminal control table line entry (TCTLE) for the line on which the error occurred. The TACLE contains all the error information necessary for correct evaluation of the error, plus special action flags that can be manipulated to alter the error correction procedure.

Terminal abnormal condition program

After the TACLE has been established, a task that executes DFHTACP is attached by the terminal control program and is provided with a pointer to the real line entry (TCTLE) on which the error occurred. After performing basic error analysis and establishing the default actions to be taken, DFHTACP gives control to DFHTEP by issuing a program control LINK request. DFHTACP passes the address of the TACLE so that DFHTEP can examine the error and provide an alternative course of action.

After DFHTEP has performed the desired function, it returns control to DFHTACP by issuing a program control RETURN request. DFHTACP then performs the actions dictated by the action flags within the TACLE, and the error-handling task terminates.

Notes:

1. You should consider prevention of data security violation. For example, if a terminal is put out of service for some time or until the cause of the failure is removed, the original operator may no longer be present, although the signon information will still be in the TCTTE when the terminal is put back into service. (See also the bulleted list on page 100.)
2. If DFHTACP has more than eight errors on a line before action can be taken, the line will be put out of service to avoid system degradation.

Terminal error program

The terminal error program analyzes the cause of the terminal or line error that has been detected by the terminal control program. The CICS-supplied version (the sample terminal error program, DFHXTEP) is designed to attempt basic and generalized recovery actions. A user-written version of this program can be provided to handle specific application-dependent recovery actions. The user-written terminal error program is linked-to in the same way as the CICS-supplied version by the terminal abnormal condition program. Equally, information relating to the error is carried in the terminal abnormal condition line entry (TACLE).

The macros and operands that are provided for generating the sample terminal error program are described in the sections that follow. The main steps are generating the sample DFHTEP module and tables by means of the DFHTEPM and DFHTEPT macros, respectively. You can select the appropriate options in this sample program, and you can base your own version on it.

There is a description of the CICS-supplied sample terminal error program (DFHXTEP), and advice on how to generate a user-written version later in this chapter.

Terminal abnormal condition line entry (TACLE)

The terminal abnormal condition line entry (TACLE) is the basic interface used by the sample DFHTEP and should be used by a user-written DFHTEP to determine the nature of the error that occurred and to indicate what course of action should be taken.

Before giving control to DFHTEP, DFHTACP establishes which default actions should be taken, depending upon the particular error condition that has been detected. The default actions are indicated by appropriate bit settings in the 1-byte fields of the TACLE labeled TCTLEECB+1 and TCTLEECB+2. The default actions and bit settings are listed in the *CICS/MVS Problem Determination Guide*.

Note: For a detailed discussion of these action bits, and the dummy terminal indicator, see the discussion under "User-written terminal error programs" on page 94. The write-abend bit (X'01' in TCTLEECB+1) is always set with the abend-task bit (X'04') as part of action 3, but both bits are suppressed if "dummy terminal" is indicated.

The code indicating the particular error condition detected is passed to DFHTEP in the 1-byte field of the TACLE labeled TCTLEPFL. These DFHTACP message codes, error codes, conditions, and DFHTACP default actions are also listed in the *CICS/MVS Problem Determination Guide*.

A format description of the terminal abnormal condition line entry (TACLE) DSECT is provided under "User-written terminal error programs" on page 94.

The sample terminal error program

CICS provides a sample terminal error program that can be used as a generalized program structure for handling terminal errors.

The source code form of the sample TEP is DFHXTEP. After DFHXTEP has been assembled, it is then link-edited as DFHTEP.

You can generate and use the sample terminal error program with the default options provided, or you can customize the terminal error support to the needs of the operating environment by selecting the appropriate generation options and variables. Because each error condition is processed by a separate routine, you can replace a CICS-provided routine with a user-written one when the sample TEP is generated.

Components

The sample terminal error program consists of the terminal error program itself and two terminal error program tables:

- The TEP error table
- The TEP default table.

Both tables contain "threshold" limits defined for the various error conditions to be controlled and accounted for by the sample DFHTEP. A "threshold" limit may be thought of as the number of error occurrences that are permitted for a given type of error on a given terminal before the sample DFHTEP accepts the DFHTACP default actions. Optionally, the number of occurrences can be controlled and accounted for over prescribed time intervals (for example, if more than three of a given type of error occur in an hour, the terminal will be put out of service).

TEP error table

The TEP error table maintains information about errors that have occurred on a terminal. The table consists of two parts (depicted in Figure 4 on page 74):

- TEP error table header (TETH) — contains addresses and constants related to the location and size of the TEP error table components.

- Terminal error blocks (TEBs) — these can be either:
  - Permanent (P-TEBs), each associated with a particular terminal;

    or
  - Reusable (R-TEBs), not permanently associated with any particular terminal.

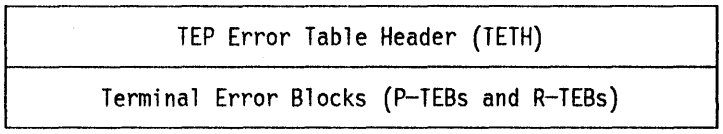

TEP Error Table Header (TETH)
Terminal Error Blocks (P-TEBs and R-TEBs)

Figure 4. TEP error table

TEBs maintain error information associated with terminals. You must specify the total number of TEBs to be generated. The maximum number needed is one per terminal. In this case the TEBs are permanent.

You can reduce the total amount of storage used for TEBs by allocating a pool of reusable TEBs, that are not permanently associated with a particular terminal. Reusable TEBs are assigned dynamically upon the first occurrence of an error associated with a terminal, and are released for reuse when the appropriate error processor places the terminal out of service.

Note: It is your responsibility to ensure that the pool is large enough to accommodate the maximum number of terminals for which errors are expected to be outstanding at any particular time. If the pool limit is exceeded, handling of terminal errors may become intermittent. **No warning is given of this condition**.

You should permanently assign TEBs for terminals that are critical to the network. For the remainder of the network, you can generate a pool of reusable TEBs.

Each TEB currently in use or permanently reserved contains the symbolic terminal identification assigned to the terminal and one or more error status elements (ESEs) as shown in Figure 5.

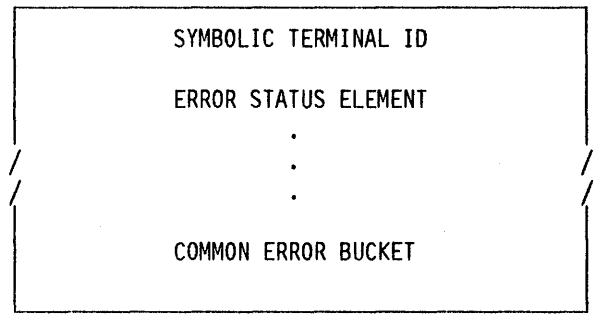


Figure 5. Terminal error block (TEB)

An ESE records the occurrence of a particular type of error associated with the terminal. The contents of an error status element are described in the TEPCD DSECT (generated by the DFHTEPM TYPE=INITIAL macro) under the comment "ERROR STATUS ELEMENT FORMAT". The number of ESEs per TEB remains constant for all TEBs. You specify the number when the TEP tables are generated. If less than the maximum number of error types (26) recognized by DFHTACP is specified, one additional ESE, referred to as the common error bucket, is generated for each TEB.

You can permanently reserve ESE space in each TEB for specific error types. Those not permanently reserved are considered reusable, and are assigned dynamically upon the first occurrence of a particular error type associated with the terminal. If an error type occurs that is not currently represented by an ESE, and if all reusable ESEs are assigned to other error types, the occurrence of this error is recorded in the common error bucket. DFHTACP can recognize far more error types than can occur in a typical terminal network. By specifying less than the maximum and allowing the sample DFHTEP to assign ESEs dynamically, you can minimize the table size, and still control and account for the types of errors relevant to the network.

Terminal error program default table

The terminal error program (TEP) default table contains the threshold limits for each type of error to be controlled and accounted for. An index array at the beginning of the default table serves a dual function. If the value in the index is positive, the error code has a permanently defined ESE in each TEB and the index value is the displacement to the reserved ESE. If the index value is negative, an ESE must be assigned dynamically from a reusable ESE if one has not already been created by a prior occurrence. The complement of the negative index value is the displacement to the threshold limits for the error type retained in the TEP default table.

Description of the sample terminal error program

The structure of the sample terminal error program (DFHXTEP) can be broken into six major areas as follows:

- Entry and initialization
- Terminal identification and error-code lookup
- Error processor selection
- Error processing execution
- Exit
- Common subroutines.

These areas are described in detail in the sections that follow.

Figure 6 on page 78 gives an overview of the structure of the sample terminal error program.

Entry and initialization

Upon entry, the sample TEP establishes base registers and addressability to the various control blocks needed to process the error (TACLE, TCTTE, TEP tables). If time support has been generated, an interval control request is issued to time-stamp the error for subsequent processing. The first entry into the sample TEP after the system was initialized causes the TEP tables to be initialized.

Terminal identification and error-code lookup

After the general entry processing, the TEP error table is scanned for a terminal error block (TEB) entry for the terminal associated with the error. If no matching entry is found, a new TEB is created. If all TEBs are currently in use (if no reusable TEBs are available), the processing is terminated and a DFHPC RETURN request is issued giving control back to DFHTACP, where default actions are taken. After the terminal's TEB has been located or created, a similar scan is made of the error status elements (ESEs) in the TEB to determine whether the type of error currently being processed has occurred before, or if it has permanently reserved ESE space. If an associated ESE is not found, an ESE is assigned for the error type from a reusable ESE. If a reusable ESE does not exist, the error is accounted for in the terminal's common error bucket. The addresses of the appropriate control areas (TEB and ESE) are placed in registers for use by the appropriate error processor.

Error processor selection

User-specified message options are selected and the messages are written to a specified transient data destination. The type of error code is used as an index into a table to determine the address of an error processor to handle this type of error. If the error code is invalid or the sample TEP was not generated to process this type of error, the address points to a routine that optionally generates an error message and returns control to DFHTACP, where default actions are taken. If an address of a valid error processor is obtained from the table, control is passed to that routine.

Error processing execution

The function of each error processor is to determine whether the default actions established by DFHTACP for a given error, or the actions established by the error processor, are to be performed. The common error bucket is processed by the specific error processor. However, the threshold limits of the common error bucket are used in determining whether the limit has been reached. Subroutines are provided in the sample TEP to maintain count and time threshold totals for each error associated with a particular terminal to assist the error processor to make its decision. Also available are subroutines for logging the status of the error and any recovery action taken by the error processor.

You can replace any of the error processors supplied in the sample TEP with user-written ones. Register linkage conventions, error conditions, DFHTACP default actions, and sample TEP error processor actions are described in comments given in the sample DFHXTEP source listing. However, sample DFHXTEP actions, in many cases, can be altered by changing the threshold limits when generating the TEP tables.

Exit

Control is passed to this routine from each error processor. This routine determines whether the terminal is to remain in service. If the terminal is to be put out of service, the terminal error block and all error status elements for that terminal will be deleted from the TEP error table unless the terminal was defined as a permanent entry. When the terminal is placed back in service, a new terminal error block will be assigned should a subsequent error occur.

Common subroutines

A number of subroutines is provided in the sample DFHTEP for use by the error processors. Each subroutine entry has a label of the form "TEPxxxxx" where "xxxxx" is the subroutine name. All labels within a subroutine start with TEPx where "x" is the first character of the subroutine name. All subroutines are arranged within the module in alphabetical order in the subroutine section. Register conventions and use of the subroutine are given as comments at the beginning of each subroutine in the source listing. The following subroutines are available if you want to write your own error processors:

TEPACT
: Used to output the names of the action bits set by DFHTACP and the sample DFHTEP in the fields TCTLEECB+1 and TCTLEECB+2 of the TACLE if appropriate PRINT options are selected when the program is generated.

TEPDEL
: Used to delete the terminal error block and error status elements for a terminal from the TEP error table on exit from an error processor.

TEPHEXCN (Used by TEPPUTTD)
: Used to convert a 4-bit hexadecimal value to its 8-bit printable equivalent.

TEPINCR
: Used to update and test the count/time threshold totals maintained in the terminal's error status element.

TEPLOC
: Used to locate or assign terminal error blocks and error status elements for a terminal identification.

TEPPUTTD
: Used to output character or hexadecimal data to a user-defined transient data destination.

TEPTMCHK (Used by TEPINCR)
: Used to determine if the time threshold limit has expired.

TEPWGHT
: Used to update the weight/time threshold values maintained in the terminal's error status elements.

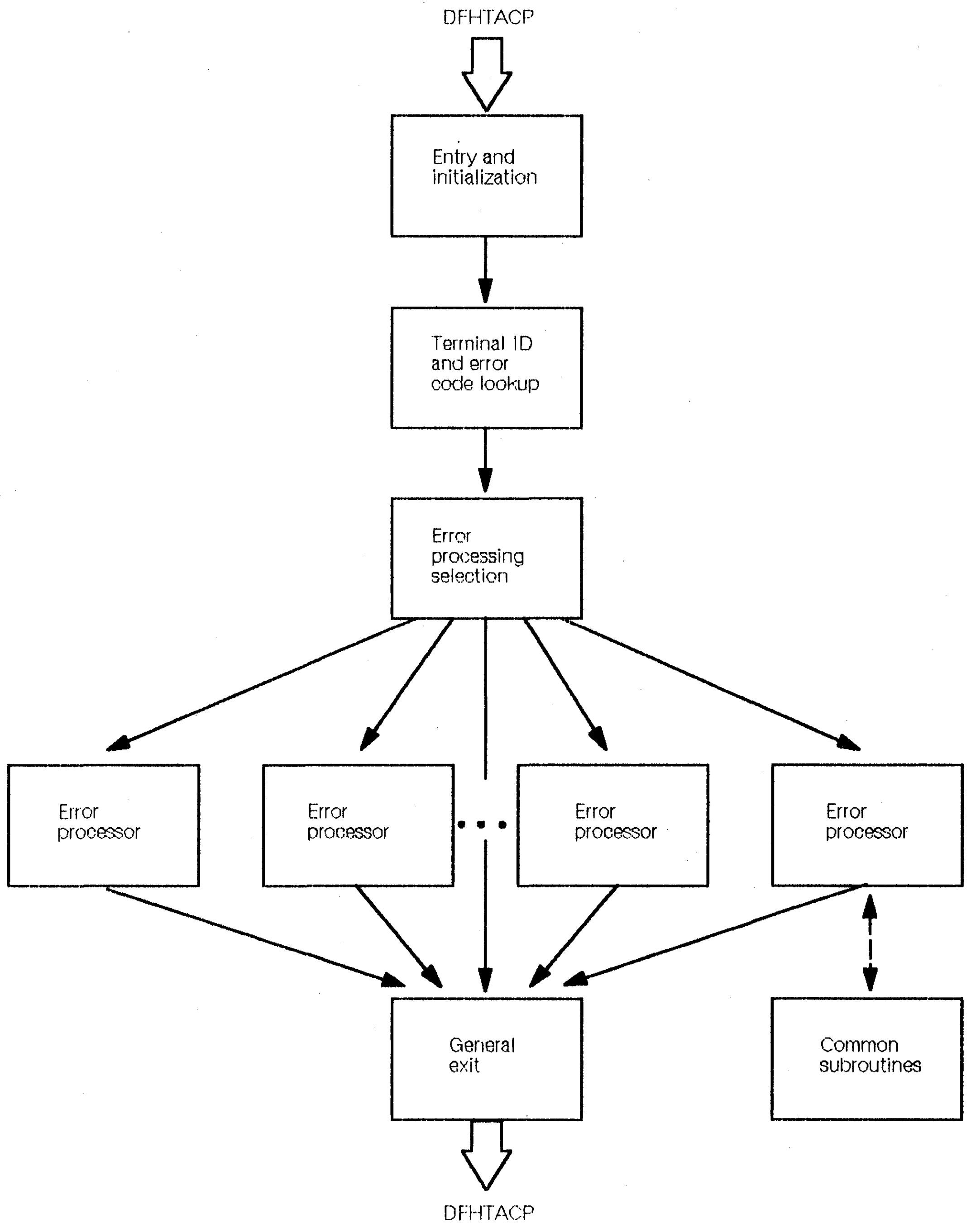


Figure 6. Sample DFHTEP overview

Sample terminal error program messages

The messages logged to the transient data destination CSMT (or, optionally, to the destination specified in the OPTIONS operand of DFHTEPM TYPE=INITIAL) are of six types, each identified by a unique message prefix. You can control the selection of each type of message using the appropriate parameters specified in the PRINT operand of DFHTEPM TYPE=INITIAL.

These messages are:

DFHTEP, ERROR – error text
During DFHTEP module generation, the PRINT parameter specified ERRORS. This message may be suppressed by using the NOERRORS option. The error text will be one of the following:

Unsupported error code, "xx"
The error code presented to DFHTEP by DFHTACP is unknown by DFHTEP.

"DFHTEPT" not defined in system.
The DFHTEP table could not be loaded into storage.

Unknown error status message, "xxxx"
The error status message presented from a remote 3270 type device could not be decoded.

None of these errors should occur.

DFHTEP, ACTION – action flag names
During DFHTEP module generation, the PRINT parameter specified TACPACTION or TEPACTION or both. If both are specified, this message is logged twice each time DFHTEP is called. The first message indicates the action flags as set by DFHTACP on entry to DFHTEP. The second message indicates the action flags as returned to DFHTACP by DFHTEP after error processing. These messages may be suppressed by using the NOTACPACTION and NOTEPACTION options.

The action flag names and descriptions are listed below. For a better understanding of the actions taken by DFHTACP, see the discussion of the TCTTEECB+1 and TCTTEECB+2 fields contained in the TACLE DSECT description in "User-written terminal error programs" on page 94.

Flag Name	Description
LINEOS	Line out of service
NO PURGE	Notpurgeable task exists on terminal
SW LINE DISCON	Switched line disconnected
DISCON SW LINE	Disconnect switched line
TERMOS	Terminal out of service
ABEND	Abend transaction
NO POLL	Take control unit off polling list
ABORTWR	Abend write request on task abend
REL TCAM TIOA	Release TCAM TIOA

DFHTEP, TID — tid

During the DFHTEP module generation, the PRINT parameter specified TID. This message contains the symbolic terminal identification of the device associated with the error. This message may be suppressed by using the NOTID option.

DFHTEP, DECB — DECB information

During the DFHTEP module generation, the PRINT parameter specified DECB. This two line message contains the DECB (printed in hexadecimal) of the terminal causing the error. The DECB is contained in the TACLE (displacement +16 [decimal]). See the TACLE DSECT described in "User-written terminal error programs" on page 94. This message may be suppressed by using the NODECB option.

DFHTEP, TACLE — TACLE information

During the DFHTEP module generation, the PRINT parameter specified TACLE. This message (printed in hexadecimal) will contain the first 16 bytes of the TACLE passed to DFHTEP by DFHTACP. See the TACLE DSECT described in "User-written terminal error programs" on page 94. This message may be suppressed by using the NOTACLE option.

DFHTEP, ESE — ESE information

During the DFHTEP module generation, the PRINT parameter specified ESE. This message contains the error status element. The message may be suppressed by using the NOESE option.

An ESE will be either 6 bytes or 12 bytes long, depending on whether the TIME option was specified when generating the TEP tables. The formats are as follows:

NOTIME

Display	Length (bytes)	
0	2	Error threshold counter or weight value in binary.
2	2	Current error count or weight value in binary.
4	1	Error code.
5	1	Not used.

TIME

Display	Length (bytes)	
0	5	Same as described in NOTIME above.
5	3	Timed threshold value in hundredths of a second.
8	4	Time of first occurrence of this error. Time given as binary integer in hundredths of a second.

Generating the sample terminal error program

The sample terminal error program and the sample terminal error table are generated by coding the DFHSG PROGRAM=CSO macro instruction in CICS system generation. The sample program and tables will provide you with default error processing for terminal errors. If you want to replace the supplied error processors with user-written error processors, you must use the DFHTEPM and DFHTEPT macro instructions to generate a sample error program and tables that include your user-written routines. Some of the parameters specified in the DFHTEPM and DFHTEPT macro instructions are related and care must be taken to ensure compatibility. The parameters concerned are identified in the descriptions of the macros later in this chapter.

If you use the sample terminal error program (DFHXTEP), you can generate the entries required in the PCT and PPT as follows:

Using CEDA
Use the CEDA INSTALL GROUP(DFHSTAND) command to obtain the PCT and PPT entries for the CSD. These entries are described below in "Using Macros".

Using Macros

1. Code the STANDARD function group in the PCT (DFHPCT TYPE=GROUP, FN=STANDARD) to get the PCT entries for the terminal error task (transaction CSTE). The TWASIZE for this transaction includes the 20 fullwords (80 bytes) required by DFHTEP in addition to the storage required by DFHTACP.
2. Code the STANDARD function group in the PPT (DFHPPT TYPE=GROUP, FN=STANDARD) to get the PPT entries for the program DFHTACP, the program DFHTEP, and the table DFHTEPT.

Job control for generating the sample terminal error program

The generation of the sample terminal error program consists of two separate assembly and link-edit steps, one to create the sample TEP module itself, and the other to create the TEP tables. See the information on the preparation of application programs in the *CICS/MVS Operations Guide* for the job control statements necessary to assemble and link-edit these components. The names under which the components must be link-edited are:

DFHTEP	Sample TEP module, assembled from DFHXTEP
DFHTEPT	Sample TEPT table, assembled from DFHXTEPT.

Generate the sample DFHTEP module — DFHTEPM macro

The sample DFHTEP module is generated by the following macro instructions:

- DFHTEPM TYPE=INITIAL — to control the printing of CICS DSECTs, provide optional routines, and indicate the type of information to be logged when errors occur.
- DFHTEPM TYPE=ERRPROC — to allow you to replace the error processors supplied with the sample terminal error program with user-written versions.
- DFHTEPM TYPE=ENTRY — to code a user "ENTRY" exit.

- DFHTEPM TYPE=EXIT – to code a user "EXIT" exit.
- DFHTEPM TYPE=FINAL – to indicate the end of the sample DFHTEP module.

```
DFHTEPM  TYPE=INITIAL
         [,DSECTPR={YES|NO}]
         [,OPTIONS=([TD|(TD,destid)|NOTD]
                   [,3270R|,NO3270R]
                   [,7770|,NO7770]
                   [,EXITS|,NOEXITS]
                   [,TIME|,NOTIME]
                   [,TCAM|,NOTCAM])]
         [,PRINT=([ERRORS|NOERRORS]
                   [,TACPACTION|,NOTACPACTION]
                   [,TEPACTION|,NOTEPACTION]
                   [,TID|,NOTID]
                   [,DECB|,NODECB]
                   [,TACLE|,NOTACLE]
                   [,ESE|,NOESE])]
```

TYPE=INITIAL
Establishes the beginning of the generation of the sample DFHTEP module itself.

DSECTPR={YES|NO}
Is used to control the printing of CICS DSECTs on the sample DFHTEP assembly listing. Its purpose is to reduce the size of the listing. The default is DSECTPR=YES.

YES
Means that printing of the DSECTs will be allowed.

NO
Means that printing of selected CICS DSECTs will be suppressed. This parameter should not be used under Assembler F.

OPTIONS=optional-routines
Is used to include or exclude optional routines in the DFHTEP module. The parentheses are required even when only one option is specified. If this operand is omitted, all default options are generated. Valid options are:

TD or (TD, destid) or NOTD
Is used to specify whether information regarding the errors is to be written to a transient data destination.

TD Means the transient data output routine is to be generated. The implied transient data destination is CSMT.

(TD, destid)
Means the transient data output routine is to be generated. The messages are sent to the destination specified by "destid," which must be defined in the destination control table.

NOTD
Means no messages are to be written to a transient data destination.

3270R or NO3270R
Is used to specify whether optional remote 3270 support is to be included.

<u>3270R</u>
Means remote 3270 errors are to be supported. More specifically, error codes 89 and 9D are supported. If you wish to supply your own error processor routines for these codes, you must specify 3270R, or make it the default.

NO3270R
Means no remote 3270 support is to be generated.

7770 or NO7770
Is used to specify whether optional 7770 support is to be included.

<u>7770</u>
Means 7770 errors are to be supported. More specifically error code 8A is supported. If you wish to supply your own error processor routine for this code, you must specify 7770, or make it the default.

NO7770
Means no 7770 support is to be generated.

EXITS or NOEXITS
Is used to specify whether "ENTRY" and "EXIT" user exit support is to be included.

EXITS
Means that branches will be taken to ENTRY and EXIT exit routines before and after error processing. Dummy exits are provided if user exits are not used.

<u>NOEXITS</u>
Indicates that no branches will be taken to user exit routines.

TIME or NOTIME
Is used to specify whether threshold limit tests are to be controlled over prescribed time intervals. An example might be putting a terminal out of service if more than three instances of a given type of error occur in one hour. The parameter must be the same as the OPTIONS operand in the DFHTEPT TYPE=INITIAL macro instruction.

<u>TIME</u>
Means this type of "threshold" testing is to be supported.

NOTIME
Means this type of "threshold" testing is not to be generated.

TCAM or NOTCAM
Is used to specify whether optional TCAM support is to be included.

TCAM
Indicates that TCAM error code '9F' is to be supported.

NOTCAM
Indicates that TCAM error code '9F' is not supported.

PRINT=print-information
Is used to specify which types of information are to be logged to the transient data destination each time an error occurs. If NOTD is specified on the OPTIONS operand, all PRINT parameters default to NO. All PRINT parameters require the transient data output routine. The parentheses are required even when only one parameter is specified.

ERRORS or NOERRORS
Is used to specify whether unprocessable conditions detected by the sample DFHTEP are to be recorded on the transient data destination.

ERRORS
Means error messages are to be logged.

NOERRORS
Means no error messages are to be logged.

TACPACTION or NOTACPACTION
Is used to specify whether DFHTACP default actions are to be recorded on the transient data destination.

TACPACTION
Means the default actions are to be logged.

NOTACPACTION
Means no default actions are to be logged.

TEPACTION or NOTEPACTION
Is used to specify whether the actions selected as a result of sample DFHTEP processing are to be recorded on the transient data destination.

TEPACTION
Means the final actions are to be logged.

NOTEPACTION
Means no final actions are to be logged.

TID or NOTID
Is used to specify whether the symbolic terminal identification of the terminal associated with an error is to be recorded on the transient data destination.

TID
Means the terminal identification is to be logged. This is the default parameter.

NOTID
Means no terminal identifications are to be logged.

DECB or NODECB
Is used to specify whether the DECB of the line associated with error is to be recorded on the transient data destination.

<u>**DECB**</u>
Means the DECB is to be logged. The hexadecimal representation of the DECB is logged as two 24-byte messages. This is the default parameter.

NODECB
Means no DECB logging is to occur.

TACLE or NOTACLE
Is used to specify whether the TACLE prefix is to be recorded on the transient data destination.

<u>**TACLE**</u>
Means the 16-byte TACLE prefix as received from DFHTACP is to be logged. This is the default parameter.

NOTACLE
Means no TACLE prefix logging is to occur.

ESE or NOESE
Is used to specify whether the ESE associated with the error is to be recorded on the transient data destination.

<u>**ESE**</u>
Means the ESE, after being updated, and before being deleted (if the action puts the terminal out of service) is to be logged. This is the default parameter.

NOESE
Means no ESE logging is to occur.

Error processor source

The sample DFHTEP provides guidance on how to prepare error processor routines, particularly with regard to register and subroutine linkage conventions. The routines must also observe the following restrictions:

- The error processor must be coded in assembler language.
- The first executable statement in the routine must be labeled TEPCDxx, where "xx" is the error code specified in the DFHTEPM TYPE=ERRPROC,CODE=errcode macro instruction, which follows.
- In addition to the register usage conventions and restrictions stated in the sample DFHTEP source, the contents of registers 12 and 13 (TCA and CSA base registers) must not be changed. The sample DFHTEP executes as a group of non-terminal-dependent tasks under CICS, and each has its own TCA during the processing of each terminal error.
- The error processor must exit to the sample DFHTEP symbolic label TEPRET.

The macro instruction required for a user "ENTRY" exit is:

```
DFHTEPM  TYPE=ENTRY
```

This macro must be immediately followed by user "ENTRY" exit code, starting with the label "TEPENTRY" and ending with a BR 14 instruction.

The macro instruction required for a user "EXIT" exit is:

```
DFHTEPM  TYPE=EXIT
```

This macro must be immediately followed by user "EXIT" exit code, starting with the label "TEPEXIT" and ending with a BR 14 instruction.

Replace error processors — DFHTEPM TYPE=ERRPROC

The macro instruction necessary to replace error processors supplied with the sample DFHTEP with user-written error processors is as follows:

```
DFHTEPM  TYPE=ERRPROC
         ,CODE=errcode
         (followed by the appropriate error
          processor source statements)
```

TYPE=ERRPROC
Indicates that a CICS-supplied error processor routine is to be replaced with the user-written error processor that immediately follows the macro instruction. This macro instruction is optional; if used, the macro must follow the DFHTEPM TYPE=INITIAL macro. One DFHTEPM TYPE=ERRPROC macro must precede each user-written error processor source routine.

CODE=errcode
Is used to identify the error code assigned to the appropriate error condition. These codes are listed in the section headed "Format description of TACLE DSECT" on page 97. For example, the 7770 timeout error condition would be entered as CODE=8A.

End of sample DFHTEP module — DFHTEPM TYPE=FINAL

The macro instruction to terminate the sample DFHTEP module is:

```
DFHTEPM  TYPE=FINAL
```

This is followed by an END DFHTEPNA statement.

DFHTEPM macro examples

1. The following is an example of the minimum number of statements required to generate a sample DFHTEP module:

   ```
   DFHTEPM    TYPE=INITIAL
   DFHTEPM    TYPE=FINAL
   END DFHTEPNA
   ```

   This example generates a sample DFHTEP module with CICS-supplied error processors and all default options. This is equivalent to the sample terminal error program obtained by coding the DFHSG PROGRAM=CSO macro instruction in CICS system generation.

2. The following is an example of a more tailored sample DFHTEP module:

   ```
   * MODULE SPECIFICATIONS

    DFHTEPM     TYPE=INITIAL,                           *
                OPTIONS=((TD,TEPQ),NO7770,EXITS),       *
                PRINT=(NOTEPACTION,NOTACPACTION),       *
                DSECTPR=NO

   * USER-SUPPLIED ERROR PROCESSORS

      DFHTEPM     TYPE=ERRPROC,CODE=81

   TEPCD81    DS  0H
              -
              -   error processor "81" source statements
              -
              -
              B   TEPRET

      DFHTEPM     TYPE=ERRPROC,CODE=9C

   TEPCD9C    DS  0H
              -
              -   error processor "9C" source statements
              -
              -
              B   TEPRET

   * USER "EXIT" EXIT CODE

      DFHTEPM      TYPE=EXIT

   TEPEXIT    DS      0H
              -
              -
              -
   Additional user source statements to be executed after
   error processing:

              -
              -
              -
   ```

```
               BR  R14

         * CONCLUDE MODULE GENERATION

            DFHTEPM     TYPE=FINAL
            END DFHTEPNA
```

In this example no 7770 support is generated, but remote 3270 support and time interval "threshold" testing support are provided. All default types of information except for TACP and TEP actions are to be logged to the TEPQ transient data destination. The CICS DSECTs will not be printed on the sample DFHTEP assembler listing. You have supplied two error processor routines (codes 81 and 9C respectively).

Generate the sample DFHTEP tables — DFHTEPT

The following macro instructions are required to generate the terminal error program tables:

- DFHTEPT TYPE=INITIAL — to establish the control section.
- DFHTEPT TYPE=PERMTID — to define permanently reserved terminal error blocks (TEBs) for specific terminals.
- DFHTEPT TYPE=PERMCODE|ERRCODE — to define permanently reserved error status elements (ESEs).
- DFHTEPT TYPE=BUCKET — to account for specific error conditions to be accounted for in the common error bucket.
- DFHTEPT TYPE=FINAL — to end the set of DFHTEPT macros.

Control section — DFHTEPT TYPE=INITIAL

The DFHTEPT TYPE=INITIAL macro instruction necessary to establish the control section for the TEP tables is:

```
DFHTEPT  TYPE=INITIAL
         ,MAXTIDS=number
         [,MAXERRS={26|number}]
         [,OPTIONS={TIME|NOTIME}]
```

TYPE=INITIAL
Establishes the beginning of the generation of the TEP tables.

MAXTIDS=number
Is used to specify the total number of permanent and reusable terminal error blocks to be generated in the TEP error table. Permanent entries are defined by the DFHTEPT TYPE=PERMTID macro instruction described later in this section. Any entries not defined as permanent will be reused when the terminal is taken out of service, or will be deleted at the request of an error processor. If an error occurs, and no TEB space is available, the error is not processed, and DFHTACP default actions are taken. The minimum number of blocks is 1. A maximum number is not checked for but should be no greater than the number of terminals in your network. **This parameter is required.**

MAXERRS=26|number

Is used to specify the number of errors to be recorded for each terminal. This value determines the number of permanent and reusable error status elements in each TEB. The maximum number that may be specified is 26. (These is also the default value.) If more are requested, only the maximum will be generated. If fewer are requested, one extra ESE will be generated for each TEB. The extra ESE is the common error bucket. Permanently reserved ESEs are defined by the DFHTEPT TYPE=PERMCODE macro instruction described later in this section. Any ESEs not defined as permanent will be dynamically assigned upon the first occurrence of a non-permanent error type associated with the terminal. By defining a number less than the maximum, and allowing the sample DFHTEP to dynamically assign ESEs, you can minimize the size of the table and still control and account for the error types relevant to the network. The minimum number that can be specified is zero. In this case only a common error bucket will be generated.

OPTIONS={TIME|NOTIME}

Is used to specify whether time threshold space is to be reserved in support of the TIME option specified in the DFHTEPM TYPE=INITIAL macro instruction. The default is OPTIONS=TIME.

TIME

Means time threshold space will be reserved.

NOTIME

Means time threshold space will not be reserved.

Define terminal error blocks — DFHTEPT TYPE=PERMTID

The DFHTEPT TYPE=PERMTID macro instruction to define permanently reserved terminal error blocks for specific terminals is as follows:

```
DFHTEPT  TYPE=PERMTID
         ,TRMIDNT=name
```

TYPE=PERMTID

Defines permanently reserved terminal error blocks for specific terminals. Permanent TEBs are defined for terminals that are critical to system operation to ensure that error processors will always be executed in the event of errors associated with that terminal. If no permanent TEBs are to be defined this macro instruction is not required. A separate macro instruction must be issued for each permanently reserved TEB. The maximum number of permanent TEBs is the number specified in the MAXTIDS operand of the DFHTEPT TYPE=INITIAL macro instruction.

TRMIDNT=name

Is used to provide the 1-to 4-character symbolic terminal identification for a permanently defined TEB. Only one terminal may be specified in each macro.

Define error status elements — DFHTEPT TYPE=PERMCODE|ERRCODE

The DFHTEPT TYPE=PERMCODE|ERRCODE macro instruction is used to change the default threshold constants of the sample DFHTEP, and to define permanently reserved error status elements:

```
DFHTEPT  TYPE={PERMCODE|ERRCODE}
         ,CODE={errcode|BUCKET}
         [,COUNT=number]
         [,TIME=(number{,SEC|,MIN|,HRS})]
```

TYPE={PERMCODE|ERRCODE}
Identifies whether the error code specified in the macro instruction is to have a permanently reserved or a dynamically assigned ESE. These macros are required only if permanently reserved ESEs are to be defined, or if the sample DFHTEP default threshold constants are to be overridden. These are listed in Figure 7 on page 92.

PERMCODE
Identifies the error code specified as having a permanently reserved ESE. Each permanently reserved ESE must be identified by a separate DFHTEPT TYPE=PERMCODE macro instruction. All DFHTEPT TYPE=PERMCODE macros must precede all DFHTEPT TYPE=ERRCODE macros.

ERRCODE
Indicates that the error code specified does not require a permanently reserved ESE, but that the sample DFHTEP default threshold constants are to be changed. Each error code requiring a threshold constant change, other than those defined as permanently reserved, must be identified by a separate DFHTEPT TYPE=ERRCODE instruction. All DFHTEPT TYPE=ERRCODE macros must follow all DFHTEPT TYPE=PERMCODE macros.

CODE={errcode|BUCKET}
Identifies the error code referred to by the TYPE=PERMCODE|ERRCODE parameter. These codes are listed in the section headed "Format description of TACLE DSECT" on page 97. For example, the 7770 time-out error condition would be entered as CODE=8A. CODE=BUCKET is only applicable to the DFHTEPT TYPE=ERRCODE macro instruction. It is used to override the default threshold constants established for the common error bucket.

COUNT=number
May be used in either the DFHTEPT TYPE=PERMCODE or TYPE=ERRCODE macro instruction to override the sample DFHTEP default threshold count limits (see Figure 7 on page 92). When the number of occurrences of the error type specified reaches the threshold limit, an error processor will normally take a logic path that causes DFHTACP default actions to be taken. If the number of occurrences is less than the threshold limit, the error processor will normally take a logic path that overrides the DFHTACP default actions. The updating and testing of the current threshold counts are

normally performed by a DFHTEP subroutine that sets a condition code that the error processor can test to determine whether the limit has been reached. If you specify 0 as the number in the COUNT operand, you will not be told when the threshold limit is reached.

TIME=time options
May be used in either the DFHTEPT TYPE=PERMCODE or TYPE=ERRCODE macro instructions to override the sample DFHTEP default threshold time limits (see Figure 7 on page 92). This parameter is only applicable when the OPTIONS=TIME parameter is specified in both the DFHTEPM and DFHTEPT TYPE=INITIAL macro instructions. When the number of occurrences reaches the threshold limit specified in the COUNT=parameter (above) within the interval specified in this parameter, an error processor would normally take a logic path that would cause DFHTACP default actions to be taken. If the number of occurrences within the interval is less than the threshold limit, the error processor normally takes a logic path that overrides the DFHTACP default actions. If the time interval has expired, the sample DFHTEP subroutine that normally updates and tests the current threshold count resets the occurrence counts, and establishes a new expiration time. In this case, the condition code set by the subroutine indicates that the threshold limits had not been reached. Time control in the sample DFHTEP starts with the first occurrence of an error type. Subsequent occurrences of the same error type **do not** establish new starting times, but are merely accounted for as having occurred within the interval started by the first occurrence. This continues until an error count reaches the threshold limit within the interval started by the first occurrence, or until the interval has expired. In the latter case, the error being processed becomes a first occurrence, and a new interval is started. A time interval of 0 means that the number of occurrences is to be accounted for and controlled without regard to a time interval. Zero is the implied time interval if the COUNT=parameter is 0 or 1. It is also the implied time interval if the time options are not generated.

The time interval may be expressed in any one of four units; hours, minutes, seconds, or hundredths of a second. This allows you to express fractional parts of a unit as whole units at a lower level. As an example, 1-1/2 minutes could be expressed as 90 seconds, or even 9000 hundredths of a second. The maximum interval must be the equivalent of less than 24 hours. While the smallest interval that can be expressed is 1 hundredth of a second, a practical minimum would be 1 to 2 minutes. This allows for access method retries and the time required to create the task required to service each error. The four methods of expressing the threshold time interval are:

number
Expresses the interval in 1/100th-second units. Parentheses are not required if this method is used. The maximum number must be less than 8 640 000 (24 hours).

(number,SEC)
Expresses the interval in whole seconds and must be enclosed in parentheses. The maximum number must be less than 86 400 (24 hours).

(number,MIN)
Expresses the interval in whole minutes, and must be enclosed in parentheses. The maximum number must be less than 1440 (24 hours).

(number,HRS)
Is used to express the interval in whole hours, and must be enclosed in parentheses. The maximum number must be less than 24 hours.

The following table illustrates the sample terminal error program default threshold count limits referred to in the TYPE, COUNT, and TIME operands of the DFHTEPT TYPE=PERMCODE|ERRCODE macro instruction.

CODE=	COUNT=	TIME=	CODE=	COUNT=	TIME=
81	3	(7,MIN)	91	0	0
84	1	0	94	7	(10,MIN)
85	1	0	95**	0	0
86	1	0	96	2	(1,MIN)
87***	50*	0	97**	0	0
88	1	0	98	5	(5,MIN)
89	100*	(7,MIN)	99	1	0
8A	2	(2,MIN)	9B	1	0
8B**	0	0	9C	5	0
8C	1	0	9D	0	(5,MIN)
			9E	0	0
8D	1	0	9F**	0	0
8E	1	0	A0**	0	0
8F	1	0	A1**	5	0
90	0	0	BUCKET	5	(5,MIN)

Figure 7. Sample DFHTEP threshold default limits

* Error processor uses a threshold "weight" instead of a threshold count (see sample DFHTEP source listing on page 102).

** Error processor maintains error count only. DFHTACP default actions are always taken regardless of the threshold limits.

*** For TCAM conditions without TACP defaults, TEP retries five times and releases TIOA. Otherwise the default TACP actions are taken.

Note: Threshold values are ignored for unit checks on local terminals (error code X'94'), and on switched lines (error codes X'94' and X'96') when they have been physically disconnected by BTAM.

Account for specific error conditions — DFHTEPT TYPE=BUCKET

The macro instruction is used to ensure that specific error conditions are always accounted for in the common error bucket:

```
DFHTEPT  TYPE=BUCKET
         ,CODE=errcode
```

TYPE=BUCKET
Generates the macro to account for specific error conditions. If MAXERR=26 is specified in the DFHTEPT TYPE=INITIAL macro instruction, this macro instruction is invalid. This macro is only required if no error codes are to be specifically accounted for in the common error bucket. Each error code must be specifically identified by a separate DFHTEPT TYPE=BUCKET macro instruction.

CODE=errcode
Identifies the error code to be specifically accounted for in the common error bucket. The error code must not be specified in the DFHTEPT TYPE=PERMCODE or TYPE=ERRCODE macro instruction.

Terminate DFHTEPT macro — DFHTEPT TYPE=FINAL

The DFHTEPT TYPE=FINAL macro instruction terminates the generation of the DFHTEP tables.

```
DFHTEPT  TYPE=FINAL
```

DFHTEPT macro examples

1. The following is an example of the minimum number of statements required to generate the TEP tables:

   ```
   DFHTEPT    TYPE=INITIAL,MAXTIDS=10
   DFHTEPT    TYPE=FINAL
   END
   ```

   This example generates 10 reusable terminal error blocks, each capable of accounting for the maximum number of error types. Time threshold control is supported, and all threshold values are the defaults supported by the sample DFHTEP. This is equivalent to the sample terminal error program table obtained by coding the DFHSG PROGRAM=CSO macro instruction in CICS system generation.

2. The following is an example of a customized TEP table (continuation character omitted):

```
* TABLE SPECIFICATIONS

       DFHTEPT     TYPE=INITIAL,MAXTIDS=10,
                   MAXERRS=5

* PERMANENT TERMINAL DEFINITIONS

       DFHTEPT     TYPE=PERMTID,TRMIDNT=TM02

* PERMANENT ERROR CODE DEFINITIONS

       DFHTEPT     TYPE=PERMCODE,CODE=81
       DFHTEPT     TYPE=PERMCODE,CODE=87,
                   COUNT=2,TIME=(1,MIN)

* OTHER THRESHOLD OVERRIDES

       DFHTEPT     TYPE=ERRCODE,CODE=BUCKET,
                   COUNT=3,TIME=(3,MIN)

* CONCLUDE TABLE GENERATION

       DFHTEPT     TYPE=FINAL
       END
```

This example generates 10 terminal error blocks, one of which is reserved for the terminal whose symbolic identification is TM02, and the other nine being reusable. Each TEB has space for five error status elements plus a common error bucket. Of the five ESEs, two are reserved for error codes 81 and 87; the remaining ESEs are available to be assigned dynamically. The threshold limits for error code 87 and the common error bucket are being changed. No specific error code is to be accounted for in the common error bucket.

User-written terminal error programs

A user-written terminal error program may be generated. The user-written DFHTEP then replaces the sample TEP provided with the DFHSG PROGRAM=CSO macro. The user-written DFHTEP will receive control as described at the start of this chapter, and therefore should use the TACLE as its basic interface with DFHTACP.

There are some situations in which CICS may attempt to send a message to an input-only terminal; for example, an invalid transaction identification message, or a message erroneously sent by an application program. You should provide a terminal error program to reroute these messages to a system destination such as CSMT or CSTL or other destinations by means of transient data or interval control facilities.

A similar situation can exist when a message is sent to a 3735 terminal operating as an input batch device. An attempt to write to the 3735 before the receipt of the end of transmission (EOT) gives control to DFHTACP. If no DFHTEP is provided, the current transaction will abend and the line will be disconnected.

Addressing the contents of the TACLE

When DFHTEP receives control from DFHTACP, the TCA facility control address (TCAFCAAA) contains the address of a TACLE. The TACLE is created by the terminal control program when the error occurs, and contains all the I/O error information provided by BTAM or TCAM.

To address the contents of the TACLE, the user-written terminal error program should contain the statements "COPY DFHTACLE" and "COPY DFHTCTLE" in that order. These define the complete DFHTCTLE DSECT. The symbolic names in this DSECT are used to address fields in both the TACLE and the real line entry associated with the error.

The TACLE consists of a 16-byte prefix (defined by "COPY DFHTACLE") and a further 48-byte section, which is a modified copy of the DECB of the real line entry at the time the TACLE was created.

To address the TACLE, the user-written terminal error program should contain the statements:

```
COPY DFHTACLE
COPY DFHTCTLE

L TCTLEAR,TCAFCAAA         POINT TO TACLE
USING DFHTCTLE,TCTLEAR
```

Note that fields normally part of the real line entry DECB have offsets increased by 16 in the TACLE.

The following fields in the DECB copy in the TACLE do **not** represent data copies from the real line entry:

```
TCTLEDCB                (Offset 24 in TACLE,
                         8 in real TCTLE)
```

This field in the TACLE points to the real line entry, whereas in the real line entry it points to the BTAM DCB for the line group.

```
TCTLEECB+1              (Offsets 17, 18 in TACLE,
TCTLEECB+2               1,2 in real TCTLE)
```

These fields in the TACLE are used as interface bytes for the terminal abnormal condition program.

```
TCTLECB+3               (Offset 19 in TACLE,
                         3 in real TCTLE)
```

This is used in the TACLE for BTAM return code for rejected I/O requests.

```
TCTLECSW                (Offsets 46, 48 in TACLE,
TCTLEALP                 30, 32 in real TCTLE)
```

These are used in the TACLE for SAM error information, apart from their normal use for BTAM lines.

Given addressability to the TACLE, you can also address the real line entry (for example, to inspect data not in the DECB copy) by coding:

```
L TCTLEAR,TCTLEDCD

USING DFHTCTLE + TCTLEECB,TCTLEAR
```

Note: The real line entry storage definition starts at TCTLEECB, 16 bytes after TCTLEPSA, and continues beyond the DECB end (TCTLESI).

To revert to addressing the TACLE, you should recode:

```
L  TCTLEAR,TCAFCAAA          POINT TO TACLE
USING DFHTCTLE,TCTLEAR       ADDRESS TACLE
```

You should take particular care that the correct addressability is established when referencing fields in the DFHTCTLE DSECT.

Note: In programs that do not require a reference to the TACLE, the following statements give direct addressability to the **real** line entry:

```
COPY DFHTCTLE
COPY DFHTCTTE

L      TCTTEAR,TCAFCAAA      POINT TO TCTTE
L      TCTLEAR,TCTTELEA      POINT TO TCTLE
USING  DFHTCTLE,TCTLEAR      ADDRESS TCTLE
```

In this case the TACLE prefix is not mentioned, and DSECT DFHTCTLE begins with field TCTLEECB.

After you have carried out the required functions and, optionally, altered the default actions scheduled by DFHTACP, the user-written DFHTEP must return control to DFHTACP by issuing the program control RETURN request. DFHTACP then performs the actions specified in the TACLE and causes the error processing task to terminate.

Format description of TACLE DSECT

TERMINAL ABNORMAL CONDITION LINE ENTRY

Dec	Hex	◄— 4 BYTES —►			
0	0	TCTLEPSA STORAGE ACCOUNTING AREA			
4	4	TCTLEPCH ADDRESS OF TRANSIENT DATA OUTPUT AREA			
8	8	TCTLEPFL ERROR FLAGS	TCTLEPF2 SPECIAL IND	NOT USED	
12	C	TCTLEPTE TCTTE ADDRESS			
16	10	TCTLEECB BEGINNING OF DECB	ACTION FLAGS	RESERVED FOR DFHTACP	BTAM/TCAM RETURN CODE
20	14				
24	18	TCTLEDCB ACTUAL LINE ENTRY ADDRESS			
28	1C				
44	2C	NOT USED		TCTLECSW BSAM STATUS	
48	30	TCTLEALP BSAM SENSE			
60	3C	TCTLEOA			

Displacement					
Dec	**Hex**	**Code**	**Bytes**	**Label**	**Meaning**
0	0		4	TCTLEPSA	Storage accounting
4	4		4	TCTLEPCH	Pointer to 100 bytes of user storage that can be used to write to transient data (first 8 bytes reserved for storage accounting). This storage must not be freed by DFHTEP, as DFHTACP may reuse it.
8	8		1	TCTLEPFL	Error flags
		81			Message too long
		83		TCEMCAAR	2740-2 auto output request (used by DFHTACP, not passed to DFHTEP)
		84			TCT search error
		85			Invalid write
		86			Polling list error
		87			Unsolicited input
		88			Input event rejected
		89			Status message received
		8A			7770 32-second timeout
		8B			Hardware buffer exceeded
		8C			Output event rejected
		8D			Output length of zero
		8E			No output area
		8F			Output area exceeded
		94			Unit check
		95			Unit check (should not occur)
		96			Unit exception
		97			Unit exception (should not occur)
		98			Negative response
		99			Undetermined I/O error
		9B			Copy error (3270)
		9C			Invalid message block
		9D			Incomplete message
		9E			No printer available for 3270 print request
		9F			Invalid destination (TCAM)
		A0			Invalid read
		A1			Invalid disconnect
		. . . . .	(All codes not listed are reserved)		
9	9		1	TCTLEPF2	Special indicator dummy terminal
		01			dummy terminal

Displacement

Dec	Hex	Code	Bytes	Label	Meaning
12	C		4	TCTLEPTE	Address of terminal entry for terminal in error
16	10		4	TCTLEECB	DECB/copy of line when error occurred
60	3C		4	TCTLEOA	For TCAM lines only. Address of the line I/O area containing the input or output message, or zero if none available.

TACLE action and information bits

The following definition of the DECB area includes TCTLEECB + 1 which contains the action bits (0, 3, 4, 5, 6, and 7) and information bits (1 and 2). This is the only portion of the copy of the DECB that can be altered. These bits are located at label TCTLEECB + 1.

Displacement

Dec	Hex		Bytes	Label	Meaning
17	11		1	TCTLEECB+1	Interface byte
		Bit 0		0...	Place line in service
				1...	Place line out of service
		Bit 1			Information bit
				.0..	Not used
				.1..	Notpurgeable task exists on terminal
		Bit 2			Information bit
				..0.	Not used
				..1.	Switched line has been disconnected by BTAM
		Bit 3		...0	Do not disconnect line
				...1	Disconnect line
		Bit 4		.... 0...	Place terminal in service
				.... 1...	Place terminal out of service
		Bit 5		.... .0..	Do not abend task
				.... .1..	Abend task
		Bit 6		.... ..0.	Leave terminal's associated control unit on poll list
				.... ..1.	Take terminal's associated control unit off poll list
		Bit 7		0	Do not abend WRITE or free terminal storage on task abend, or no task present on terminal
				1	Abend terminal WRITE requests and free terminal storage on task abend or no task present on terminal

18	12		1	TCTLEECB+2	Interface byte 2
		Bit 0		1...	Release TCAM incoming message
		Bits 1-7			Reserved
19	13		1	TCTLEECB+3	BTAM return code
24	18		4	TCTLEDCB	Actual line entry address
46	2E		2	TCTLECSW	BSAM status
48	30		1	TCTLEALP	BSAM sense

The following factors should be considered when altering the action bits in the TACLE:

- For TCAM unsolicited input errors with either the terminal out of service or in receive-only state, a loop will occur if the default action of purging the incoming message does not occur and the status of the terminal is not altered.
- The dummy terminal indicator at TCTLEPF2 is set on errors such as: (1) BTAM return on input, (2) binary synchronous outputs performed for TCP where no terminal is indicated, and (3) other errors from which no specific terminal is indicated. Therefore, if a dummy terminal is indicated, task abend and write abend are not set (see below). The dummy terminal is only used to identify the line.
- The switched-line disconnected bit (X'20' at TCTLEECB+1) is used by DFHTACP upon return from DFHTEP to logically disconnect (by issuing a WRITE BREAK) the switched line that has been physically disconnected by BTAM. If DFHTEP determines that the line has not been physically disconnected, DFHTEP may reset this bit. DFHTCP can communicate this disconnect condition for BISYNC lines to DFHTACP by setting the bit TCBSWB in the field TCTLEDI in the real line entry. DFHTCP will do this when a READ INITIAL or READ CONNECT completes with an I/O error or when a mandatory disconnect sequence (DLE-EOT) is received from the remote terminal. OS/VS BTAM may DISABLE a switched line and convey this fact by setting the bit X'08' in the field TCTLEES. This flag may be tested in the TACLE.
- The disconnect switched-line bit (X'10' at TCTLEECB+1) is used by DFHTEP to request that DFHTACP actually makes the disconnection (by means of a WRITE DISCONNECT).
- If the switched-line disconnected bit or the disconnect switched-line bit is on, upon return from DFHTEP, the task abend bit should also be set to purge the task from the disconnected terminal. If this is the case and if the task is not purgeable from the terminal, DFHTACP writes an INTERCEPT REQUIRED message to destination CSMT and places the terminal out of service.
- The abend transaction bit (X'04' in TCTLEECB+1) is always associated with two other bits as part of TACP action 3. These other bits are notpurgeable task and write abend (X'40' and X'01' respectively, both in TCTLEECB+1).
- Write abend is always set on at the same time as abend transaction. It has the effect of clearing the TCTTE of the original write request indicators, if the error being processed occurred on a TC WRITE.
- Notpurgeable task is set to on if a transaction is currently associated with the terminal, but if this transaction ID was specified with TPURGE=NO.

- None of the abend task, write abend, or notpurgeable task bits will be set if the dummy terminal indicator is on, even if DFHTACP would normally set default action 3 (abend transaction) for the error being processed. So, the following remarks apply only to errors related to a real terminal.
- Abend task has no effect if no transaction is associated with the terminal, except in the case where a pseudoconversational task has been associated with this terminal. In this case, the next transid (TCTTETC) will be cleared. Otherwise, if notpurgeable task is indicated, the transaction remains attached to the terminal (normally in SUSPEND state) and DFHTACP writes the DFH2522 INTERCEPT REQUIRED message to CSMT; if the transaction is not marked notpurgeable, it is abended with code ATAI, or rarely, ATAD.
- Write abend has no effect if the TCTTE was associated with a READ request. In this case the normal result will be that, if the line and terminal remain in service, the read will be retried.

Example of a user-written terminal error program

The following is an example of the logic steps necessary to design a portion of the terminal error program, called the "DFHTEP recursive retry routine." In this example, 10 retries are provided for each terminal; however, the logic could be used for any number of retries. The following assumptions are made:

USER FIELD A
(PCISAVE)
Represents a 6-byte field in the process control information (PCI) area of the TCTTE (see the TCT macro definition of the TCTUAL operand). This field is used to preserve the count of input and output from the TCTTE when the first error occurs. These counts are contained in 3-byte fields located at TCTTENI and TCTTENO within the TCTTE.

USER FIELD B
(PCICNT)
Represents a user-defined field used to accumulate the count of recursive errors. It should be in the process control information (PCI) area of the TCTTE.

SYSTEM COUNT
(TCTTENI)
Represents the 6-byte field in the TCTTE that contains the terminal input and output counts (TCTTENI+TCTTENO). In the example, these two adjacent fields are considered as one 6-byte field.

Because this example requires access to the TCT terminal entry (TCTTE) to examine the SYSTEM COUNT and to locate the process control information (PCI) area, the DFHTCTTE symbolic storage definition is included so that fields may be symbolically referenced.

DFHTEP recursive retry routine

```
***********************************************************************
*                                                                     *
*                   DFHTEP RECURSIVE RETRY ROUTINE                    *
*                                                                     *
***********************************************************************
TEPBAR  EQU   2                      TEP PROGRAM BASE
TCTTEAR EQU   9                      BASE REGISTER FOR TCTTE
PCIBAR  EQU   8                      BASE FOR PCI
        DFHTCA                       TASK CONTROL AREA
        COPY  DFHTCTTE               COPY TCTTE DEFINITION
        EJECT
        COPY  DFHTACLE               COPY TACLE SYMBOLIC DEFINITIONS
        COPY  DFHTCTLE               COPY DECB DEFINITION
        EJECT
PCIAREA DSECT
PCISAVE DS    6X                     USER FIELD A
PCICNT  DS    PL2                    USER FIELD B
        EJECT
DFHTEP  CSECT
        BALR  TEPBAR,0               ESTABLISH PROGRAM ADDRESSABILITY
        USING *,TEPBAR
        L     TCTLEAR,TCAFCAAA       LOAD TACLE ADDRESS
        L     TCTTEAR,TCTLEPTE       LOAD TCTTE BASE WITH
*                                    TCTTE ADDRESS
        L     PCIBAR,TCTTECIA        LOAD PCI AREA ADDRESS
        USING PCIAREA,PCIBAR         ESTABLISH ADDRESSABILITY
        TM    PCICNT+1,X'0C'         HAS USER FIELD B EVER BEEN
*                                    INITIALIZED TO A PACKED
*                                    DECIMAL NUMBER?
        BO    CKCOUNT                .. YES, SO COMPARE THE
*                                    SYSTEM COUNT WITH THE
*                                    EXISTING COUNT IN FIELD B;
RESET   MVC   PCICNT,=PL2'+0'        .. NO, SO INITIALIZE FIELD
*                                    B TO A PACKED DECIMAL 0.
        PCISAVE DS XL6               SAVE THE CURRENT SYSTEM
        MVC PCISAVE(L'PCISAVE),TCTTENI
*                                    COUNTS.
THIS IS A NEW
*                                    ERROR, OR FIRST TIME THROUGH
INCR    AP    PCICNT,=P'1'           INCREMENT THE NUMBER OF
*                                    TIMES THIS SAME ERROR HAS
*                                    OCCURRED.
(RECURSIVE COUNT)
        CP    PCICNT,=P'10'          HAS THE MAXIMUM RECURSIVE
*                                    ERROR LIMIT BEEN REACHED?
        BNE   RETRY                  .. NO, SET ACTION
*                                    INDICATORS FOR RETRY ATTEMPT
        ZAP   PCICNT,=P'0'           *  CLEAR AND RESET USER FIELDS
        PCISAVE DS XL6               *  FOR NEXT ERROR SET
        MVC PCISAVE(L'PCISAVE),TCTTENI
        B     NORETRY                ACTION INDICATORS FOR NO-RETRY.
CKCOUNT PCISAVE DS XL6               HAS SYSTEM COUNT CHANGED SINCE
```

```
         CLC PCISAVE(L'PCISAVE),TCTTENI
*                                           LAST ENTRY TO TEP?
         BNE   RESET                        .. YES; THAT MEANS THIS IS
*                                           A NEW ERROR SINCE SOME I/O
*                                           ACTIVITY HAS OCCURRED ON
*                                           TERMINAL
         B     INCR                         .. NO; THAT MEANS THIS IS A
*                                           RECURSIVE ERROR, SO
*                                           INCREMENT THE RECURSIVE COUNT
*                                           AND CHECK FOR RETRY.
RETRY    DS    0H                           THE USER WOULD INCLUDE HERE
         .                                  THE CODE NECESSARY TO ALTER
         .                                  THE FLAGS IN THE TACLE SO
         .                                  THAT A RETRY CAN BE PERFORMED
                                            ON THE TERMINAL.
NORETRY  DS    0H                           THE USER WOULD INCLUDE HERE
         .                                  THE CODE NECESSARY TO ALLOW
         .                                  DFHTACP TO TAKE FINAL ACTION
         .                                  ON THE TERMINAL (I.E., ABEND
         .                                  TASK, PUT LINE OUT OF SERVICE,
                                            ETC.)
         LTORG
         END
```

The above example is intended only to serve as an illustration of a recursive error handling technique and of the steps necessary to establish addressability to the applicable control blocks.

Note: You will probably wish to prevent data security violation. This may happen, for example, when a terminal has been put out of service and the operator leaves it. The master terminal may put that terminal back into service, and another operator may use it with the original operator's security key. You can provide automatic sign-off by including the following code in the DFHTEP (for example, after the label "NORETRY", which appears in the example above):

```
LR    R3,R1
L     R4,TCAFCAAA
ST    TCTTEAR,TCAFCAAA
L     R1,TCTTELEA
DFHPC TYPE=LINK,PROGRAM=DFHSFP
LR    R1,R3
ST    R4,TCAFCAAA
```

After providing addressability to every terminal entry, similar actions may be performed for every terminal on a line that is taken out of service.

User-written actions for particular cases

This section provides guidance on how to write your own terminal error program to handle error conditions from several devices. The following topics are discussed:

- Switched BSC temporary text delay (TTD)
- 7770 32-second timeout
- 2740 Model 2

- Teletypewriter (countries outside the US only)
- 3270 unavailable printer
- 3600 BSC
- 3275 dialed timeouts
- 3270 locked buffer.

Switched BSC temporary text delay (TTD)

When a temporary text delay indication is received, BTAM, after retrying the operation up to seven times, will turn on TCTLESF7 (TCTLESF=X'01') and return control to CICS indicating that an error has occurred. CICS will then invoke DFHTEP for error analysis.

BTAM may also turn on TCTLESF7 when a data record ending with ENQ is received (the terminal detected a parity or transparency error). Therefore, DFHTEP should also examine the I/O area pointed to by TCTLEIOA to determine if it contains STX ETX (TTD) or EOT ...data... ENQ.

7770 32-second timeout

If a terminal connected to the 7770 Audio Response Unit goes on hook while no I/O operation is outstanding, the 7770 does not present the unit exception to the channel. This situation can occur when the terminal operator makes an inquiry and hangs up before receiving a response. After this occurs, all writes to the line appear to complete normally. All reads complete normally at the end of the 32-second timeout with a zero data length.

When a 32-second timeout occurs, either the terminal operator has not entered anything for 32 seconds, or the terminal operator has hung up and the 7770 did not inform CICS. CICS cannot distinguish between these two conditions; therefore, CICS handles every 32-second timeout as an error condition. DFHTACP goes to DFHTEP with defaults of DISCONNECT SWITCHED LINE and ABEND THE TRANSACTION. If DFHTEP does not disconnect the switched line, CICS writes the "ready" message and initiates another read.

2740 model 2

When DFHTACP detects a negative response from a 2740 Model 2, the write operation will be retried after a 10-second time delay if the user-written TEP has been coded to retry the write. This delay allows for operator reaction time, and other factors. If the delay time factor is to be changed, this may be done by storing the new time delay factor at TCTTEBC. The value is a positive binary number representing hundredths of a second (10 seconds would have a value of F'1000' or X'000003E8'), which is calculated by adding the delay value to the value contained in CSACSCC. The cause of the negative response may be determined by examining the field TCTLERSP. The contents of TCTLERSP and the meaning of each follow:

X'04'	Terminal in bid mode
X'02'	Terminal in communicate mode
X'20'	Terminal in communicate mode with document device down
X'10'	Terminal in local mode
X'13'	Terminal in communicate mode but out of paper
X'08'	Contents of buffer are being printed.

Caution: If you do not set a long enough time delay, this may cause a loop at the terminal, and the reset key will not be recognized if it is pressed.

Teletypewriter (countries outside the US only)

There are no default actions provided by DFHTEP for the teletypewriter (countries outside the US only). You have to handle all exceptional conditions. In the case of an ID error or a severe transmission error, you may want to abend the task and disconnect the line so that the computer can accept a new connection. Under these circumstances, it is recommended that, after entry to DFHTEP, the interface byte for the status of the task, line, and terminal, (that is, TCTLEECB + 1) be set to the following values before returning to DFHTACP:

```
Bits    0 1 2 3 4 5 6 7
Values  0                 Line in service
          *               Unchanged
            0             Switched-line disconnect request "off"
              1           Disconnect line
                0         Terminal in service
                  1       Abend task
                    *     Unchanged
                      *   Unchanged
```

3270 unavailable printer

This condition arises when a print request is made through the 3270 print request facility and there are no printers on the control unit, or the printer(s) is in one of the following conditions:

- Out of service
- A task is currently attached
- Currently busy on a previous operation
- Intervention required.

The terminal control program recognizes this condition and issues a READ BUFFER operation to collect the data into a line I/O area (LIOA). The LIOA is of the same format as a terminal I/O area (TIOA) would be if an application program had issued a terminal control read buffer request. Thus, the TIOA DSECT may be used to reference the LIOA.

The TCP then obtains a TACLE, and attaches DFHTACP with the error code X'9E' (TCEMCUP). The TACLE fields relevant to this situation are:

TCTLEIOA	Pointer to the LIOA
TCTLETLA-1	Pointer to first printer on control unit or zero (no printers).

DFHTACP writes the DFH2508 UNAVAILABLE PRINTER message to the CSMT destination and LINKs to DFHTEP with no default actions set.

On return from DFHTEP, DFHTACP will perform the following actions, based on the field TCTLETLA in the TACLE:

1. If TCTLETLA-1 is all FFs (−1 set by DFHTEP) DFHTACP assumes that DFHTEP has disposed of the data to be printed and requires the keyboard of the originating terminal to be restored.
2. If TCTLETLA-1 is 0 (zero) DFHTACP will assume that no printer is available, and the keyboard of the originating terminal will not be restored.
3. If TCTLETLA-1 is neither 0 (zero) nor −1 (all FFs) DFHTACP assumes that TCTLETLA-1 is the address of a printer. An interval control PUT command will be performed to the provided terminal. The transaction to be initiated is CSPP (print program), and the time interval will be zero. If CSSP is to be scheduled, AUTOTRN=YES must be specified in DFHSG PROGRAM=TCP to include the AVAIL logic.
   a. If an error occurs on the interval control PUT, DFHTACP will write the DFH2531 IC FAILURE message to the destination CSTL (DFHTACP error code X'91'). DFHTACP will then link to TEP again with the high order bit (X'80') set in the TACLE field, TCTLETLA-1, and the IC error value from the TCA field TCAICTR will be placed into the TACLE field, TCTLEECB+3. This is done in order for TEP to have a last chance to dispose of the data. On the second RETURN from TEP to DFHTACP, DFHTACP will reexamine TCTLETLA-1. If TCTLETLA-1 is -1 (all FFs), DFHTACP will restore the originating terminal's keyboard. Otherwise, the keyboard will remain locked.
   b. If no error occurred on the interval control PUT command, DFHTACP will check for the following printer conditions:
      1) Out of service
      2) Intervention required
      3) Other than RECEIVE or TRANSCEIVE status.

      If one of these conditions is true, DFHTACP will issue the DFH2532 PRINT QUEUED message to the destination CSMT (DFHTACP error code X'90').
4. DFHTACP will then terminate any PRINT requests on the originating terminal, free the LIOA, and perform normal action flag processing on the originating terminal.

Note that all scheduling and error handling for 3270 printers operating under TCAM is provided by the message handler.

3600 BSC

There is no special default processing provided in DFHTEP for BTAM-supported 3600 BSC terminals.

3275 dialed timeouts

CICS will always disconnect a 3275 switched line if there is no activity on the line for two minutes.

In some countries, the local laws prohibit a switched line from remaining connected for more than 30 seconds if no line activity is taking place. This action is initiated by the modem (for example, model 3976-3 for countries other than the U.S.A.) dropping out of its ready state and going on hook. In this situation, CICS will disconnect the line, and manual intervention may be required to reestablish the connection.

3270 locked buffer

To prevent data displayed on a 3270-system display unit from being copied to a 3270-system printer, the display unit buffer can be locked by placing a protected alphanumeric attribute byte (BIT2=1, BIT3=0) in address 0. This will cause any attempt to use the copy command to end with sense status X'C4C1'. For further information, see the appropriate 3270 Information Display System manual.

Chapter 2.10. The node error program

As with the terminal error program for non-VTAM devices, the node error program (NEP) for VTAM-attached terminals is available in three forms:

- The dummy node error program
- The CICS-supplied sample node error program
- User-written versions.

All three types are discussed in the following sections.

Notes:

1. Node error programs, not terminal error programs, must be used for terminals and logical units supported via the ACF/VTAM interface.
2. In this chapter, "VTAM 3270" refers to the non-SNA 3270 connected through VTAM, and "3270 compatibility mode" refers to an SNA 3270* connected through VTAM.
3. If you code an EXEC CICS HANDLE CONDITION TERMERR command in your application program, it is ***sometimes*** possible for the application program to handle exceptional cases, rather than using a node error program. The TERMERR condition is driven if DFHZNAC actions an ABTASK(ATNI abend). Note that TERMERR is application-related, and is not an alternative to the node error program, which must be used for session-related problems. Dealing with errors in the application program is particularly useful in an intersystem communication (ISC) environment.

This chapter has two reading sequences. The first is a general overview of the use of the node error program. It begins at "Background to CICS-VTAM error handling" on page 110. The second, which begins at "When an abnormal condition occurs" on page 117, goes into more detail about:

- CICS components involved in an abnormal condition
- Logging
- The sample NEP
  - Components
  - Routing
  - Options
  - Macros for its generation.
- User-written NEPs
  - Multiple NEPs
  - Macros for generation
  - Error processors.
- 3270 unavailable printer
- Session failures.

* IBM Trademark. For a list of trademarks see page iii.

If you are new to the node error program, you should read the first sequence before you look at the more detailed sections that follow. If you are familiar with NEPs, you will often be able to go straight to the second sequence and look at the sections that particularly interest you.

Background to CICS-VTAM error handling

- In general, errors detected in CICS-VTAM terminal control are queued for handling by a special task, the CICS node error handler (transid 'CSNE'). In addition, CICS finds it convenient to use the same technique for certain housekeeping work, such as sending "good-morning" messages, and logging session starts and ends, which are not "errors" at all.
- In a few cases, exceptions signalled to CICS by VTAM are not treated as errors, and are not passed to the node error handler. For example, CICS often sends an SNA BID command as part of automatic transaction initiation. Rejection of the BID with exception code 0813 (wait) is a standard response, and CICS handles the retry in terminal control without calling this an "error".

  For the rest of this description, only the errors are considered.
- The CSNE task runs as a "background" task. "Background" means that it is not associated with any one CICS terminal. At any time, there will be at most one such task, working on the single node error queue.

  All node errors on the queue are analyzed in turn by a table-driven, CICS-supplied program called DFHZNAC (node abnormal condition program). It is not intended that you should ever modify this. However, it is sometimes helpful to understand the internal logic of DFHZNAC. (For example, when DFHZNAC does not call DFHZNEP.)
- DFHZNAC will LINK to a module called DFHZNEP (if present in the CICS system) when processing most node errors. The interface for this LINK is described in "When an abnormal condition occurs" on page 117.

  This formal ZNAC/ZNEP interface gives you the opportunity to supply your own code to analyze error conditions, change default actions, and take additional actions specific to your applications.
- The key features of the DFHZNAC-DFHZNEP interface are as follows:
  - DFHZNEP must be written as an assembler program.
  - It is LINKed-to separately for each error on the queue.
  - Communication between the two modules is through fields in the transaction work area (TWA) for the CSNE task.

  On each DFHZNEP invocation, one such field contains a 1-byte internal error code, assigned by DFHZNAC, which identifies the type of error. Other fields identify the CICS TCTTE (LU) associated with the error, and any SNA sense codes. There are also fields for ZNEP to pass back user messages for subsequent logging by ZNAC.

Further fields contain "action flags". Each flag represents an action that DFHZNAC may take when DFHZNEP returns control to DFHZNAC. These "actions" are of different types:

- Reporting (error messages, dumps of control blocks, actions taken)
- Status changes (for example, of TCTTE)
- Clean-up work (cancel any associated transaction, end the VTAM session).

Some of these flags are for information only. One informs you that the VTAM session either has just failed or will be terminated unconditionally. But most of the flags can be reset by the DFHZNEP program to modify the actions taken on return to DFHZNAC. However, DFHZNAC sometimes overrules the changed flags and acts on the original flag settings. For an example of this, see Figure 8 on page 120, which shows two flags as "not resettable". If those flags have been unset, DFHZNAC disregards the change.

The detailed internal error codes used by DFHZNAC, with its default actions and messages for each case, are described in the *CICS/MVS Problem Determination Guide*.

- CICS supplies a pregenerated "dummy" DFHZNEP, which simply returns control when DFHZNAC LINKs to it. Because it leaves all action flags unchanged, the DFHZNAC default actions are not affected.

Why use a NEP to supplement CICS default action?

The following list gives some of the reasons why you might want to write your own node error programs to add to the default actions provided by CICS and VTAM.

- Not all "errors" represent communication system failures. Some errors (like trying to write zero-length data) may reflect special situations in applications, needing special action.
- There might be duplication of diagnostic information between CICS and VTAM (if a session, or, worse, the whole network control program goes down). So you might want to suppress the messages from DFHZNAC. Or, you might wish to use another technique to notify errors (DFHZNAC only writes its messages to the transient data queues CSMT and CSTE).
- In other cases, you might want to change the amount of diagnostic information produced by CICS — the default varies with the error type. For example, the VTAM RPL associated with an error may be printed when you do not want it, or not printed when you do.
- There could be application-related activity to be done when a node error occurs. For example, if a message fails to be delivered to a terminal, it may need redirecting to another. With messages sent with exception-response only, CICS may not have the data available to resend it, but the requesting application might be able to recreate it. For example, if an error were signalled during the sending of a document to a printer, it might be able to restart from the beginning, or the specific page, of a document.

- Some devices, such as the 3650 Retail Store System, return application-type data in "User Sense Data" fields. This can only be retrieved in a NEP. The NEP has to catch and save data for further application programs.
- If CLSDSTP=NOTIFY has been specified on DFHSIT, the NEP will be informed of both successful and unsuccessful VTAM CLSDST PASS requests. A user-written NEP can reestablish the interrupted session with CICS when a CLSDST PASS request fails. For more information, see "Application routing failure" on page 122.

An overview of writing a NEP

The only restriction on your DFHZNEP module is conformity to the defined interface: a LINKed-to assembler program that uses the defined TWA fields to analyze an error and then returns to DFHZNAC. The tools provided for writing node error programs are macro level, and NEPs are normally written in macro-level. The source code of the "dummy" NEP provided by CICS may be used as a skeleton on which to build a single NEP.

CICS also provides a variety of macros to help you generate more complex, "sample" NEPs. These are merely to help you develop your own NEPs — you do not have to use any of them.

Your node error handling logic may be written as a number of modules, but the one that receives control from DFHZNAC must be called DFHZNEP. Note that some of the macros described later give the option of using different names.

DFHZNEP code can use standard CICS functions (LINK, XCTL) to invoke other user modules. Each module thus requested must, of course, be defined in the CICS PPT. PPT entries are also needed for DFHZNAC and DFHZNEP themselves. These are automatically generated by the CEDA command:

```
CEDA INSTALL GROUP(DFHVTAM)
```

or by the standard PPT macro:

```
DFHPPT TYPE=GROUP,FN=VTAM
```

The dummy NEP

The dummy NEP, DFHZNEP, is supplied with CICS. It is not the same as the sample node error program. The dummy NEP prints the dummy TCTTE when the threshold of temporary VTAM storage problems has been reached. DFHZNEP then returns control to DFHZNAC. It performs no other processing.

The sample NEP

The CICS sample node error program is a generalized program structure for handling errors detected from logical units. None of its components is generated as part of the standard CICS generation process, but instead may be optionally generated as described in this section and in "The sample node error program" on page 123.

The "sample" NEPs that CICS provides are designed with two main features:

- The samples assume you will want to invoke separate routines (ERRor PROCedures, or ERRPROCs) to handle different "groups" of error types. You specify which of the DFHZNAC internal error codes are to be regarded as a "group" for processing by any one routine, and then supply the code for that routine. CICS has some standard cases to help you. More information is given about them below.
- The samples may work in association with a separately generated module called a node error table. This can be used to build up statistics for each error group that the NEP will process. This table is analogous to the terminal error table, DFHTEPT, used by the equivalent CICS-BTAM sample error program.

  Some of the CICS-supplied error processors use the node error table - the 3270 GROUP=1 is an example. (See "User-supplied error processors – DFHSNEP TYPE=ERRPROC" on page 130.) Other processors do not use a NET.

The node error table

It is easier to understand the sample NEP if you first look at the node error table structure in more detail.

Node error table is often abbreviated to NET. This can be confused with "net" as in "network", and also with NETNAME, which here means the name of a node error table, and has nothing to do with NETNAME as used in the CICS TCT.

You can generate a node error table using the CICS macro DFHSNET. See "Node error table" on page 125 and "Generating the sample node error table – DFHSNET" on page 129. You choose how sophisticated this table is to be.

The node error table must be defined in the PPT as a RESIDENT program. This makes it easy for the NEP to find it (using a CICS LOAD request), but ensures that any counters are not reset by reloading. You can give the table any name you like. The default is DFHNET.

Basically, the table comprises sets of error recording areas. Each set is called a node error block (NEB) and is used to count node errors for one LU at one time. You may dedicate certain NEBs for use with certain specific LUs throughout a CICS run; and you can leave other, reusable NEBs for general use. If you expect that you might be accumulating error statistics about 10 LUs concurrently, you would need 10-12 NEBs.

Each NEB may contain multiple recording areas, one being used for each group of errors you want to distinguish. The error "groups" correspond to those in the NEP. That is, they are groups of error types requiring separate processing logic.

Each recording area is known as an error status block (ESB). You specify the space reserved for each ESB, and it typically includes space to count the errors, or record when the first of the present series occurred. Note that in any one NEB the counting is for one LU only.

Finally, you can specify a threshold and a time limit in the table. These are simply constants, which can be used by code in the NEP to test an ESB, to see if a given type of error has occurred more than the threshold number of times in the stated interval. The time limit also affects the interval between using a general NEB for one LU and then reusing it for another.

A minimal NET would simply consist of a handful of NEBs, each with just one ESB, all types of error that are of interest being grouped together.

Coding the sample NEP

The sample NEP is coded using the macro DFHSNEP. The basic form is as follows:

```
         DFHSNEP TYPE=INITIAL
         Specific error handling code (more details below)
Example: DFHSNEP TYPE=DEF3270
         DFHSNEP TYPE=FINAL
         END     DFHNEPNA
```

By default, this generates a module called DFHZNEP, which works with a node error table called DFHNET. If you want to use another table, you could code NETNAME=MYTABLE after TYPE=INITIAL. Details of the DFHSNEP macro are given in "Generating the sample node error program" on page 126.

The best way to understand the sample code is to generate a standard NEP (try the one with TYPE=DEF3270 shown in "3270 error processors – DFHSNEP TYPE=DEF3270" on page 128 and look at the resultant assembler listing. Here is a description of the code.

The INITIAL and FINAL macros generate the basic skeleton of the NEP. This comprises some initialization code, and some common routines. All the code is built round the assumption that you have a node error table as previously described.

The initial code will first test the internal error code passed from DFHZNAC to see if it belongs to a group the NEP needs to handle. The groups are identified by the code you sandwich between the DFHSNEP INITIAL and FINAL macros. This is described in "Generating the sample node error program" on page 126. If the particular error code is not of interest to NEP, control is returned at once to DFHZNAC, to take default actions.

Otherwise, the relevant node error table is located by a CICS LOAD request. (As previously explained, this table should be resident in virtual storage.) The NEP code will then locate the correct ESB within a selected NEB. This may either be an NEB permanently dedicated to the LU in error (named NEB), or one taken from the general pool.

The initial code will then invoke the appropriate user logic for that error group. On entry, this code will then have set up any necessary pointers to the CSA, TCA, TWA, the TCTTE for the LU in error, the NEB, and the ESB. For details, see "Generating the sample node error program" on page 126.

The common routines in the NEP provide common services for your own logic. They count and time-stamp errors in the ESB, or test whether error thresholds have been exceeded. They are not documented outside the sample listings. You can generate a NEP without them if you wish.

Your own code is inserted between the DFHSNEP TYPE=INITIAL and TYPE=FINAL macros. Each section of user logic, intended to handle a particular "group" of error types, is headed by a macro of the type:

```
DFHSNEP TYPE=ERRPROC,CODE=(ab,cd,...),GROUP=n
```

where X'ab', X'cd' .... are the ZNAC internal error codes you want to process, and n is the number of the error "group", and therefore also of the corresponding ESB, within an NEB, in the node error table. Successive ERRPROCs should use groups 1, 2, 3, ...

The TYPE=ERRPROC macros serve several purposes. They:

- Inform the NEP generation how many error groups there are
- Show which error types are to be included in each group
- Introduce the code for each group.

Note that any one DFHZNAC error code should only figure in one error group, and that any code not mentioned will simply be ignored by the NEP. You follow each ERRPROC macro with your own logic. This should begin with standard code to save registers, or set up addressability, which is best copied from sample NEP listings.

To help you, CICS provides certain standard ERRPROCs to handle specific errors on two different types of LU. These are for a non-SNA 3270 (BSC 3270 attached to CICS-VTAM), and for interactive SNA logical units like a 3767. Further information is given in the second sequence of this chapter.

The code for non-SNA 3270 can be generated by coding

```
DFHSNEP TYPE=DEF3270
```

where you would otherwise code an ERRPROC macro plus following logic of your own. In effect, TYPE=DEF3270 defines two error groups, each an ERRPROC. The first group comprises the three ZNAC error codes X'D9', 'DC', and 'DD' that are associated with receipt of 3270 status codes. The second group only contains error code, X'42', corresponding to the "unavailable printer" condition, a specific CICS exception condition signalled when it cannot allocate a printer in response to a "PRTREQ" screen-copy request.

The 3270 sample code provides error handling, under VTAM, for non-SNA 3270, which is compatible with that previously provided by CICS-BTAM in the sample TEP. This sample code is not intended to cover all error conditions. Note that the code is not enough for SNA 3270 (LU session type 2). Error conditions presented for these have different ZNAC error codes and may require different handling.

You may find that the CICS-supplied code is not sufficient for other application-related reasons. Perhaps you want to try to reacquire lost sessions after a time-interval. The code supplied for the 3767 covers only one error group with one ZNAC error code, X'DC', which may occur under contention protocol.

You can use these CICS-supplied ERRPROCs to generate a valid DFHZNEP listing, for tutorial purposes, without having to write any user code.

You should be aware of the following limitations of this NEP design:

- Any error types you have not allowed for are ignored by the NEP, not accumulated into "error buckets".
- You may want to handle a particular situation whenever it arises, even though ZNAC may assign it different error codes in different situations. For example, on SNA 3270, switching in and out of TEST state, generates X'082B' status (presentation-space integrity lost). This might result in any of several ZNAC error codes.

In the sample NEP structure, you would either need to test for this last case in separate ERRPROCs, or group all the ZNAC error codes together. If you wrote your own NEP code from scratch, you would simply test the TWA field containing the status, on entry to your NEP.

Multiple NEPs

CICS allows you to define a "NEP class" for each transaction ID in the PCT and TCT (NEPCLASS option of the CEDA DEFINE PROFILE and CEDA DEFINE TYPETERM commands or NEPCLAS operand of DFHPCT TYPE=ENTRY and DFHTCT TYPE=TERMINAL macros). This is a 1-byte binary value from 1 to 255, the default being zero. The purpose of NEPCLASS is that, while a transaction is running on the LU, you can obtain a special version of node error handling, suitable for that NEPCLASS (sometimes this is called a "transaction-class error routine").

The DFHZNEP that gets control from DFHZNAC must test the NEPCLASS in effect at that time for the LU associated with the error. Then it either transfers control to a suitable module (the actual "NEP"), or branches to a specific bit of code within itself.

CICS provides a sample with the DFHZNEPI macro. See "DFHZNEPI macros" on page 133. DFHZNEPI macros generate a DFHZNEP module that is purely a routing module. This inspects the NEPCLASS in effect for the node error passed by DFHZNAC, and transfers control (LINKs) to another module, the real NEP, according to a NEPCLASS/name routing table built up by the macros.

If no NEPCLASS is in effect (equivalent to CEDA DEFINE PROFILE NEPCLASS(0) or DFHPCT TYPE=ENTRY NEPCLAS=0), or the NEPCLASS is not in the routing table, a default module is invoked. You must specify the name of this in the DFHZNEPI TYPE=INITIAL macro. (See "Default transaction-class routine — DFHZNEPI TYPE=INITIAL" on page 133.) If you do not specify the name, no module is invoked.

You now have to provide the sub-NEPs for the various NEPCLASSes, including, of course, one for the default NEPCLASS(0). Each of these sub-NEPs needs a separate PPT entry. You have the same choice in coding each sub-NEP as you had when there was just one — you can code your own, or use the CICS sample macro DFHSNEP. If you use DFHSNEP, note that there's another operand on the TYPE=INITIAL macro, NAME=, which means that the generated module can be given any name you wish (to match the DFHZNEPI routing). You can use a different node error table with each sub-NEP.

Before you start using NEP routing, consider the following:

- The association of an LU (TCTTE) with a transaction NEPCLASS is only valid for about the time that the CICS task exists. Errors detected after a CICS task has ended (for example, because of a problem with a delayed output message) may not be associated with the NEPCLASS of the creating transaction.

  Another problem can occur when CICS is about to start a new task for the LU as a result of an internal request from another CICS task (by an EXEC CICS START request, for example). This is usually called automatic transaction initiation. Before the task is started, CICS may have to open a fresh session if none exists, by issuing a VTAM SIMLOGON request, and then, as mentioned earlier, it may have to send a BID command. The intended task will not be attached until all this completes successfully. The NEPCLASS is not picked up from the PCT until then. This means that any errors arising in the ATI process (perhaps an error on BIND or BID) will occur before the NEPCLASS is correctly set, so they may get routed to the default NEP and not the one for the NEPCLASS. This complicates the total node error handling for the application.

  As an example, consider an application that contacts unattended programmable controllers overnight in order to read in the day's input. Recovery design in such an application is fundamental, and has to allow for errors both in ATI and in file transmission. To separate these into two NEPs could be an unnecessary complication.

- The extra development effort for a NEP split on a NEPCLASS basis might not be justified. Generally, if logic is to be split, it is on an LU basis (programmable controllers may be running applications other than 3270).

To conclude this overview, remember that the CICS sample NEPs are a good source of ideas for you to write your own NEPs, but they might not be the ideal framework for your particular need. It is probably a good idea for you to write straightforward NEPs at first.

When an abnormal condition occurs

The following CICS components are involved when an abnormal condition is detected from a logical unit:

- The terminal control program VTAM section — DFHZCA, DFHZCB, DFHZCC, DFHZCP, DFHZCQ, DFHZCW, DFHZCX, DFHZCY, and DFHZCZ.
- The node abnormal condition program — DFHZNAC.
- The CICS-supplied sample node error program, or your own version(s) of that program (DFHZNEP).

For logical units, all information concerning the processing state of the terminal is contained in the TCTTE and the request parameter list (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 action flags, set by DFHZNAC to assist the node error program, are in TWAOPTL, which is in DFHZNAC's transaction work area (TWA).

If you want to modify DFHZNAC's actions following an abnormal situation, DFHZNEP can interrogate TWAOPTL and modify the bit settings. If you agree with DFHZNAC's proposed actions, TWAOPTL is left unaltered.

In most cases, DFHZNEP can modify DFHZNAC's proposed actions. The only time that DFHZNAC overrides the routine's modification of TWAOPTL is when a logical unit is to be disconnected from CICS; that is, when DFHZNAC determines that the abnormal situation requires that CICS issue the ACF/VTAM CLSDST macro instruction for a logical unit. In such a case, DFHZNAC will disconnect the terminal and abnormally terminate the task even if DFHZNEP tries to block such actions.

Resetting of the task termination flag by the node error program is also ignored if a negative response has been sent to a logical unit, or if DFHZEMW is to write an error message to the logical unit.

When control is returned to DFHZNAC from DFHZNEP, DFHZNAC performs the actions specified in TWAOPTL (except when disconnecting logical units, as noted above), issuing messages and setting error codes, as necessary.

DFHZNAC assumes that system sense codes are available upon receipt of an exception response from the logical unit. Thus, analysis is performed to determine the reason for the response. Decisions, such as which action flags to set and which requests are needed, are made based upon the system sense codes received. If sense information is not available, default action flags are set, and DFHZEMW is scheduled to send a negative response, if a response is outstanding, with an error message to the terminal.

The *CICS/MVS Problem Determination Guide* lists the actions taken by DFHZNAC upon receipt of inbound system sense codes.

Before executing the specified mandatory executive routines, DFHZNAC links to DFHZNEP, where you can define the criteria upon which action can be undertaken.

You need to code a node error program only if you wish to perform additional error processing beyond that performed by DFHZNAC. DFHZNAC gives control to DFHZNEP by issuing a DFHPC LINK request. DFHZNAC also passes the address of the TCTTE concerned, so that you can specify further recovery actions based on the processing state of the logical unit. When the node error program has performed its functions, control is returned to DFHZNAC by issuing a DFHPC RETURN request.

Upon entry to DFHZNEP, the following fields are available to you:

- The error code generated by DFHZNAC. The error codes are discussed in this chapter and in the *CICS/MVS Problem Determination Guide*. The error code is located at TWAEC.
- The action flags set by DFHZNAC. These flags are shown in Figure 8 on page 120. They are defined in the *CICS/MVS Problem Determination Guide*. The collective field name for these flags is TWAOPTL.
- The address (TWATCTA) of the TCTTE.
- The terminal name, at TWANID.
- The sense codes received by DFHZNAC:
  - TWASR1 and TWASR2, system sense codes
  - TWAUR1 and TWAUR2, user sense codes.
- The flag TWAPFLG. A setting of TWAPIP indicates that a VTAM CLSDST PASS request is in progress. For more information about the use of TWAPIP, see "Application routing failure" on page 122.

Linkage to DFHZNEP is provided by CICS. Fields in the TWA are defined in the copy section DFHVTWA, which provides a DSECT of the node abnormal condition program's TWA.

Byte	Action Flag Options		
	Action-Flag Label	**Description**	
TWAOPT1	TWAOAF	Print action flags	(X'80')
	TWAORPL	Print RPL	(X'40')
	TWAOTCTE	Print TCTTE	(X'20')
	TWAOTIOA	Print TIOA	(X'10')
	TWAOBIND	Print bind area	(X'08')
TWAOPT2	TWAOAS	Abandon deferred SEND	(X'80')
	TWAOAR	Abandon VTAM RECEIVE	(X'40')
	TWAOAT	ABEND task	(X'20')
	TWAOCT	CANCEL task	(X'10')
	TWAOASM	SIMLOGON required	(X'02')
	TWAOGMM	Good morning message required	(X'08')
TWAOPT3	TWAOINT	CREATE may be set	(X'80')
	TWAONINT	NOCREATE may be set	(X'40')
	TWAONCN	Close session (not resettable by NEP)	(X'10')
	TWAOSCN	Close session (user reset allowed)	(X'08')
	TWAONEGR	SEND Negative response from TWASS1	(X'04')
	TWAOOS	Keep node out of service	(X'02')
	TWAOCN	Cancel session (abnormally) (not resettable by NEP)	(X'01')
TWAOPT4	TWANOMG1	No INITIAL ERROR message	(X'01')
	TWANOMG2	No sense-code message	(X'02')
	TWANOMG3	No VTAM 3270 message	(X'04')
	TWANOMG4	No security message	(X'08')
	TWANOMG5	No action message	(X'10')

Figure 8. DFHZNAC action-code bytes and available options

Explanation of the flags follows:

The first five labels (TWAOAF, TWAORPL, TWAOTCTE, TWAOTIOA, and TWAOBIND) are principally debugging aids. DFHZNAC writes the desired information to the CSMT log if its accompanying bit is set.

The next six labels (TWAOAS, TWAOAR, TWAOAT, TWAOCT, TWAOASM, and TWAOGMM) are task-related.

The NEP can abend the task by setting TWAOAT, or cancel it by setting TWAOCT. The difference between 'abend the task' and 'cancel the task' is that the former does not take effect until the task requests or completes a terminal control operation. The latter, however, takes effect as soon as system and data integrity can be maintained. Setting TWAOAT to 'abend the task' is normally sufficient, except where the task performs lengthy processing (such as a data base browse) between terminal requests. If both TWAOAT and TWAOCT are set, TWAOCT (cancel task) takes priority.

If the task is to be abnormally terminated, sends and receives are purged. If TWAOGMM is set, the next transid is cleared and any COMMAREA associated with the terminal is released — except in the case of permanent transids (specified on the terminal definition as TRANSACTION(name)). If the TYPETERM for the terminal indicates that GMM is supported (LOGONMSG(YES)) and TWAONINT is off, and the terminal is not in a BMS paging session, the good-morning message transaction is initiated (transaction specified by the GMTRAN SIT parameter).

The flags in byte TWAOPT3 are node-related. If TWAOCN is set, the task is abnormally terminated and communication with the node is lost.

TWAONINT forces TWAOCN.
TWAOOS forces TWAOCN.
TWAOCN forces TWAOAR, TWAOAS, and TWAOAT.
TWAONEGR forces TWAOAR and TWAOAT.
TWAOGMM is ignored if TWAONINT is set.

Notes:

1. If a definite response SEND has been performed, CICS has to issue a RECEIVE in order to obtain the response. If the response is negative, DFHZNAC will be entered and will set flags TWAOAS (abandon the SEND) and TWAOAR (abandon the RECEIVE). TWAOAR must be left on to ensure that the RECEIVE for the response is abandoned.
2. If the request is to be retried, and the break connection action flag is off (that is, if TWAOCN is off), then TWAOAS and/or TWAOAR and/or TWAONEGR must be off as well as TWAOAT.
3. The abend code returned as a result of setting TWAOCT is unpredictable.
4. TWAOGMM forces TWAOAT only if set on by the node error program.

TWAOOS or TWAONINT for an intersystem communication session causes the system entry to be put out of service if, as a result of that action, there are no allocatable sessions left.

Setting TWAOOS indicates no further processing is to be done for this node. The node is logically out of service.

Setting TWAOSCN provides the same function as TWAONCN, but you can reset it yourself if the session is not to be closed.

The flags in byte TWAOPT4 are message-related. You can set them yourself to prevent the messages indicated being issued.

If you schedule DFHZNAC because of the receipt of an exception response, the sense information in the TCTTE is available to DFHZNAC and DFHZNEP to determine any necessary actions.

If you schedule DFHZNAC 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 and transaction-class error routine analysis and processing are permitted, but you should not attempt to retry the message.

However, if the application program handles the TERMERR condition, the transaction will not be abended. Control is returned to the program. In this circumstance, no further use may be made of the failed session.

The DFHZNAC error message is sent to the master-terminal log after linking to DFHZNEP, so that you can control the printing of messages. You may also send user-written messages to the log using the transient data facility. To write your own messages, you must code the DFHTD TYPE=PUT macro instruction directly into the node error program.

The CICS terminal control macro (DFHTC CTYPE=COMMAND) enables you to issue ACF/VTAM indicators in the node error program. The functions available are explained in "Chapter 4.7. Modifying the terminal control table" on page 237.

DFHZNAC logging facility

You can use the logging facility available in DFHZNAC to aid in retrieving related information (that is TIOA, CSA, TCA) about a problem in a real-time environment.

For example, if a logical unit sends an exception response to data sent from the host during the processing day, you can examine the TIOA to locate the problem.

DFHZNEP can pass the address of the TIOA plus a desired length (not exceeding 220 bytes) in DFHZNAC's TWA. On return to DFHZNAC, the data is logged to the CSMT or CSTL transient data log for future inspection.

TWA fields are:

Name	Length	Content
TWANLD	4 bytes	Address of data to be logged or zeros
TWANLDL	2 bytes	Desired length of data to be logged

Note: No data in excess of 220 bytes is logged.

Application routing failure

The EXEC CICS ISSUE PASS command, which is described in the *CICS/MVS Application Programmer's Reference* manual, passes control from CICS to another named VTAM application. The EXEC CICS ISSUE PASS command invokes the VTAM macro CLSDST with the value OPTCD=PASS. If CLSDSTP=NOTIFY has been specified in DFHSIT, the EXEC CICS ISSUE PASS command also passes the value PARMS=(THRDPTY=NOTIFY). CICS is then notified of the outcome of any CLSDST PASS request.

This notification results in an informative message being issued, and causes DFHZNAC to invoke your NEP, whether the CLSDST request has failed or succeeded. The NEP can discover that a CLSDST PASS request is in progress by examining field TWAPFLG for the pass-in-progress indicator, TWAPIP. The success or failure of the CLSDST PASS request can be determined by examining the error code at TWAEC.

If the pass operation fails, DFHZNAC sets up a default set of recovery actions that can be modified by your NEP. A possible recovery, when, for example, the target application program is not active, would be to reestablish the session with the initial application using a SIMLOGON request and for CICS to send its "good morning" message to the terminal. The default action is to leave the session disconnected and to make it NOCREATE.

The sample node error program

The sample node error program provides a general environment for the execution of error processing routines (error processors), each of which is specific to certain error codes generated by the node abnormal condition program. Sufficient optional error processors for normal operation of VTAM 3270 or interactive logical unit networks are provided; these can be easily supplemented or replaced by user-supplied processors.

Three types of error may occur in a VTAM network:

- Errors in the host system
- Communication errors, such as session failures
- Abnormal conditions at the terminal, such as intervention required and invalid requests.

A sample node error program is supplied with CICS, and may be used as the basis of each subsequent node error program that you write. This provides you with:

- A general environment within which your error processing programs may be added
- Fundamental error recovery actions for a VTAM 3270 network that are consistent with those provided in the sample terminal error program for a BTAM 3270 network
- The default node error program in a system that has several node error programs.

The CICS-supplied sample node error program is described in greater detail below.

Compatibility with the sample terminal error program

The default error processors for VTAM 3270s in the sample node error program provide facilities for error handling similar to those for BTAM 3270s that are processed by the sample terminal error program.

Receipt of sense/status codes corresponds to error codes X'D9', X'DC', and X'DD'. Weighted counts of these messages are maintained against numeric and time thresholds. If the numeric threshold is exceeded, default actions are taken. If the time threshold is reached, the count is reset. This is equivalent to the function in the sample TEP, except that sense/status arising out of the "from" device on a COPY command is now presented to the node error program as an error on the "to" device, thus exceeding the threshold, which causes the request to be terminated, although the terminal remains in service. Some of the weights for errors that occur on the 3270 display have been revised, otherwise the weight and threshold values are the same as the defaults used in the sample TEP. Time threshold maintenance is mandatory and not optional as in the sample TEP.

For further information on time and threshold count limits, see the information on the sample terminal error program in "Chapter 2.9. The terminal error program" on page 71.

The 3270 message "unavailable printer" corresponds to error code X'42' (interval control PUT request has failed). The algorithm used for printer selection differs in VTAM support. The retry algorithm in the sample node error program is similar to this new selection algorithm.

Components

The sample node error program comprises the following components:

- The routing mechanism.
- The node error table.
- Optional common subroutines.
- Optional error processors for 3270 or optional error processor for interactive logical units. A node error program cannot be generated with both 3270 and interactive logical unit error processors.

The components are described below.

Routing mechanism

The routing mechanism invokes the appropriate error processor depending on the error code provided by the node abnormal condition program.

Groups of one or more error codes are defined in the DFHSNEP. Each group is associated with an index (in the range X'01' through X'FF') and an error processor. A translate table is generated and the group index is placed at the appropriate offset for each error code. Error codes not defined in groups have a zero value in the table. An error processor vector table (EPVT) contains the addresses of the error group processors, positioned according to their indexes. The vector table extends up to the maximum index defined; undefined intermediate values are represented by zero addresses.

On entry to the sample node error program, initialization establishes addressability to the node error table (NET) and, if included, the common subroutine vector table (CSVT). The error code is translated to obtain the error group index. A zero value causes the node error program to take no further action, otherwise the index is used to obtain the address of the appropriate error processor from the EPVT. A zero address causes the node error program to take no further action. Otherwise a call is made to the error processor. This is entered with direct addressability to the following areas: NET, TCTTE, TCA, CSA, and CSVT. When the error processor has executed, the node error program returns control to the node abnormal condition program.

Node error table

The node error program may use a node error table (NET) that comprises the node error blocks (NEBs) that are used to maintain error status information for individual nodes (see Figure 9). Some or all of the NEBs may be permanently reserved for specific nodes, others are dynamically assigned to nodes when errors occur. The latter are used exclusively for the nodes to which they are assigned until they are explicitly released. All the NEBs have an identical structure of error status blocks (ESBs). Each ESB is reserved for one error processor and associated with it by means of the appropriate error group index. The ESB length and format may be customized to the particular error processor that it serves.

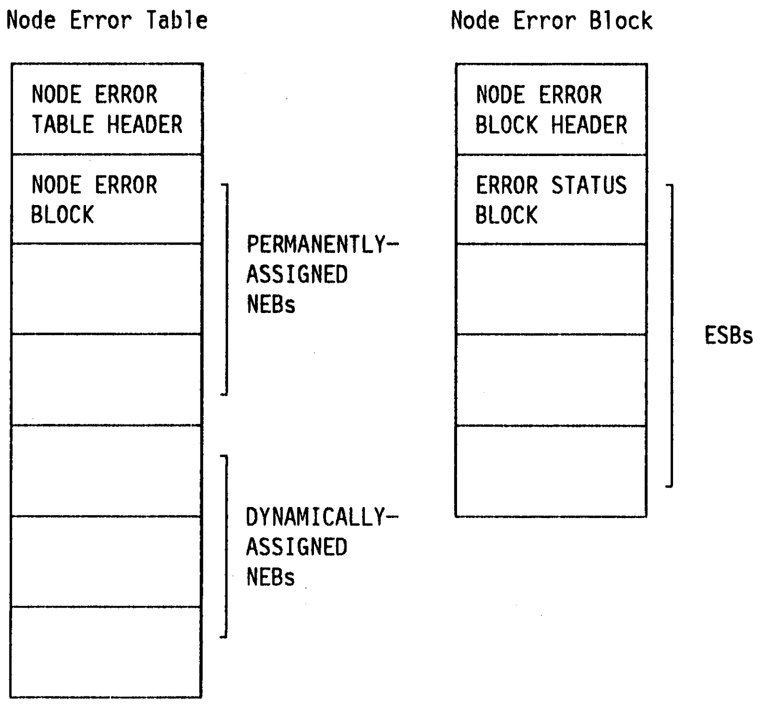


Figure 9. Format of node error table and node error block

Optional common subroutines

The common subroutines are addressed via the CSVT and provide error processors with the following functions:

1. Locate or assign NEBs and ESBs on the basis of node identification and error group index.
2. Time-stamp an error, update an error count, and test an error count against numeric and time threshold values.

3. Release a dynamically assigned NEB from a particular node.

Optional error processors for 3270 logical units

Two error processors are supplied as follows:

- Group index 1, error codes X'D9', X'DC', and X'DD'.

  These error codes correspond to the receipt of sense/status bytes in the user sense fields of the RPL. The error processor locates an ESB of the standard format and updates a weighted error count. The weight, threshold, and timer values are based on those used by the sample terminal error program for a BTAM 3270 except as noted in the previous section. If the threshold is not exceeded, the abend send, abend receive, abend transaction bits, and all the print action flags are turned off. Otherwise the default actions are taken and the NEB is released if it is reusable.

- Group index 2, error code X'42'.

  This code means that no 3270 printer was available to satisfy a PRINT request made at a 3270 screen. The error processor examines the printers defined for this screen to determine why they were unavailable. If either is busy on a previous PRINT or COPY request (that is, a task is attached with transaction identification of CSPP or CSCY) or is no longer unavailable, that printer address is returned to the node abnormal condition program which will retry the PRINT request with an IC PUT command. Otherwise the default actions are taken. (For more details, see the section "3270 unavailable printer" on page 134.)

Optional error processor for interactive logical units

- Group index 1, error codes X'DC'.

  This error code, in combination with a user sense value of X'081B', indicates a "receiver in transmit mode" condition. The action flags are manipulated in order to allow the failing SEND request to be retried.

Generating the sample node error program

The routing mechanism, common subroutines, IBM-supplied error processors, and user-supplied error processors are generated by means of DFHSNEP macros.

The sample node error program and table need to be assembled and link-edited. See the chapter on the preparation of application programs in the *CICS/MVS Operations Guide* for the job control statements required. When using the sample node error program, the CSNE TWA size supplied by CICS will be adequate.

Note that an extra 24 bytes are required for the common subroutines register save area, and further space is required for the error processor save area. The CICS sample processors use 4 bytes of this area.

The DFHSNEP macro to generate the sample node error program has five types, as follows:

TYPE=INITIAL
to generate the routing mechanism and, optionally, the common subroutines.

TYPE=DEF3270
to generate the default IBM-supplied error processors for 3270 devices.

TYPE=DEFILU
To generate the default IBM-supplied error processor for interactive logical units operating in contention mode.

TYPE=ERRPROC
To indicate the start of a user-supplied error processor.

TYPE=FINAL
To indicate the end of the sample node error program.

The order of these macro instructions is constrained so that one TYPE=INITIAL macro instruction appears first, and one TYPE=FINAL macro instruction appears last.

Routing mechanism — DFHSNEP TYPE=INITIAL

The following operands can be used on the DFHSNEP TYPE=INITIAL macro instruction:

```
DFHSNEP  TYPE=INITIAL
         [,CS=NO]
         [,NAME=name]
         [,NETNAME=netname]
```

TYPE=INITIAL
Indicates the start of the sample node error program and causes the routing mechanism to be generated.

CS=NO
Specifies that the generation of the common subroutines is to be suppressed. This operand should not be specified if TYPE=DEF3270 is included.

NAME=name
Specifies the name of the node error program module identifier. The name must be a string of one through eight characters. This operand is optional, and the default is DFHZNEP. If the interface module DFHZNEP (generated by the DFHZNEPI macro) is used, this operand must be specified (with a name other than DFHZNEP).

NETNAME=netname
Specifies the name of the node error table to be loaded at initialization. The name must be a string of 1 through 8 characters. This operand is optional, and the default is DFHNET.

3270 error processors — DFHSNEP TYPE=DEF3270

The DFHSNEP TYPE=DEF3270 macro has the following format:

```
DFHSNEP  TYPE=DEF3270
```

TYPE=DEF3270
Specifies that the IBM-supplied error processors for 3270 logical units are to be included in the node error program. This macro causes the following source code to be generated:

DFHSNEP TYPE=ERRPROC,GROUP=1,CODE=(D9,DC,DD)
Sense/status error processor code

DFHSNEP TYPE=ERRPROC,GROUP=2,CODE=42
Unavailable printer error processor code.

Error processors for INTLU — DFHSNEP TYPE=DEFILU

The DFHSNEP TYPE=DEFILU macro has the following format:

```
DFHSNEP  TYPE=DEFILU
```

TYPE=DEFILU
Specifies that the IBM-supplied error processor for interactive logical units is to be included in the node error program. This macro causes the following source code to be generated:

```
DFHSNEP TYPE=ERRPROC,GROUP=1,CODE=DC
(receiver in transmit mode error processor code)
```

Terminating DFHSNEP entries — DFHSNEP TYPE=FINAL

The DFHSNEP TYPE=FINAL macro has the following format:

```
DFHSNEP  TYPE=FINAL
```

TYPE=FINAL
Indicates the end of the node error program and causes the error processor vector table (EPVT) to be generated.

Generating the sample node error table — DFHSNET

The DFHSNET macro is used to generate a node error table. An entry for each sample node error table generated must be included in the PPT with RES=YES specified.

```
DFHSNET  [NAME=name]
         [,COUNT=threshold]
         [,ESBS=(index,length,...)]
         [,NEBNAME=(name,...)]
         [,NEBS=number]
         [,TIME=(interval,units)]
```

NAME=name
Specifies the identifier to be included in the NET header. It must be a string of one through eight characters. This operand is optional, and the default is DFHNET.

COUNT=threshold
Specifies the error count threshold that is to be stored in the NET header for use by the common subroutines to update standard ESBs. If the threshold is exceeded, the error processor that invoked the subroutine is informed by a return code. The maximum value is 32767. This operand is optional, and the default is 100.

ESBS=(index,length,...)
Specifies the ESB structure for each NEB. This operand is coded as a sublist. Each element of the sublist comprises two values; "index" specifies an error group index for which an ESB is to be included in the NEB; "length" specifies the status area length, in bytes, for that ESB. The parentheses may be omitted for a single element. Index must be specified as a 2-character representation of a 1-byte hexadecimal number in the range 01 through FF (a leading 0 can be omitted). "length" is constrained only by the fact that an 8-byte NEB header plus a 4-byte header for each ESB must be contained within the maximum NEB length of 32767 bytes. If a null value is specified, a standard ESB with a status area length of 6 bytes is assumed. This is suitable for use by the common subroutines in maintaining a time-stamped error count. This operand is optional and defaults to 1. This causes each NEB to be generated with one ESB for error group 1 with a status area length of 6 bytes.

NEBNAME=(name,...)
Specifies the names of nodes that are to have a permanently assigned NEB. The names specified are assigned, in the order specified, to the set of NEBs requested by the NEBS operand. Any remaining NEBs are available for dynamic allocation to other nodes as errors occur. The name must be a string of 1 through 4 characters. The parentheses can be omitted for a single name. This operand is optional and has no default.

NEBS=number
Specifies the number of NEBs required in the NET. The maximum valid number is 32767; the default is 10.

TIME=(interval,units)
Specifies the time interval that is to be stored in the NET header for use by the common subroutines to maintain error counts in standard ESBs. If the threshold specified in the COUNT operand is not exceeded before this time interval elapses, the error count is reset to 0. Specify "units" as SEC, MIN, or HRS. The maximum values for "interval" are as follows: (86400,SEC), (1440,MIN), or (24,HRS). This operand is optional, and the default is to (7,MIN).

Note: You can use the sample node error program above (with a name other than DFHZNEP) as a transaction-class routine for the interface module, DFHZNEPI.

User-supplied error processors — DFHSNEP TYPE=ERRPROC

The DFHSNEP TYPE=ERRPROC macro is used to indicate the start of a user-supplied error processor. The actual error processor code should immediately follow this macro. The assembly should be terminated by the statement: END DFHNEPNA.

The following operands can be used on the DFHSNEP TYPE=ERRPROC macro instruction:

```
DFHSNEP  TYPE=ERRPROC
         ,CODE=(error-code,...)
         ,GROUP=error-group-index
```

TYPE=ERRPROC
Indicates the start of a user-supplied error processor.

CODE=(error-code,...)
Specifies the error codes that make up the error group, and which are therefore handled by the error processor supplied. The operand is coded as a sublist of 2-character representations of 1-byte hexadecimal codes. (The parentheses may be omitted for a single code.) For each code specified, the error group index is placed at the equivalent offset in the translate table. Thus when this code occurs, the appropriate error processor can be identified.

GROUP=error-group-index
Specifies an error group index for the error processor. This index is used to name the error processor, locate its address from the error processor vector table (EPVT), and optionally associate it with an ESB in each NEB. The index specified must be a 2-character representation of a 1-byte hexadecimal number in the range 01 through FF (a leading zero can be omitted). The error processor name has the form NEPROCxx, where "xx" is the error group index. A CSECT statement of this name is generated, which causes the error processor code to be assembled at the end of the node error program module and to have its own addressability.

If you intend to add your own error processors to the sample node error program, you should be aware of the following conventions used by the sample node error program:

Register Assignment

Register	Use
0	Work register
1	NET base register
2	NEB base register
3	ESB base register
4	Error count increment, also work register
5	Work register
6	Work register
7	Work register
8	Work register
9	Work register
10	TCTTE base register
11	Sample node error program base register
12	TCA base register
13	CSA base register
14	CSVT base and error processor link register Common subroutine link register
15	Error processor branch register Common subroutine branch register

Notes:

1. Registers 12 and 13 must be preserved at all times.
2. Register 14 must be saved for return from error processors. The CSVT is coded after the BALR to the error processor and so this register is also the CSVT base.
3. Registers 1, 10, 12, 13, 14, and 15 are set up on entry to error processors.
4. Registers 14-11 may be saved by error processors in an area reserved in the TWA at label TWAEPRS. Registers 15-11 do not need to be restored before return from error processors.
5. Registers 4-9 may be saved by common subroutines in an area reserved in the TWA at label TWACSRS. They must be restored before return from the subroutines.

DSECTs

The following DSECTs are provided:

Node error table header: This contains the table name and common information relevant for all the node error blocks (NEBs) in the table.

```
DFHNETH    DSECT
NETHNAM    DS      CL8          Table name
NETHNBN    DS      H            Number of NEBs in table
NETHNBL    DS      H            Length of NEBs in table
NETHTIM    DS      BL4          Error count time interval
NETHECT    DS      H            Error count threshold
NETHFLG    DS      X            Flag byte
NETHINI    EQU     X'01'        Table initialized
           DS      X            Reserved
NETHFNB    DS      0F           First NEB
```

Node error block: The table contains node error blocks that are used for recording error information for individual nodes. These may be permanently assigned to specific nodes or dynamically assigned at the request of error processors.

```
DFHNETB    DSECT
NEBNAM     DS      CL4          Node name
NEBFLG     DS      X            Flag byte
NEBPERM    EQU     X'01'        Permanently assigned NEB
           DS      XL3          Reserved
NEBFESB    DS      0X           First NEB
```

Error status block: The NEBs may contain error status blocks. These are reserved for specific error processors and are identified by the corresponding error group index. An ESB may have a format defined by you, or may have a standard format suitable for counting errors over a fixed time interval.

```
DFHNETE    DSECT
ESBEGI     DS      X            Error group index
ESBFLG     DS      X            Flag byte
ESBSTAN    EQU     X'01'        Standard format ESB
ESBTTE     EQU     X'02'        Time threshold exceeded
ESBCTE     EQU     X'04'        Count threshold exceeded
ESBSLEN    DS      XL2          Status area length
ESBHLEN    EQU     *-DFHNETE    ESB header length
ESBSTAT    DS      0X           Status area
```

The following fields apply to the standard format:

```
ESBTIM     DS      BL4          Time stamp
ESBEC      DS      XL2          Error count
```

Common subroutine vector table: The CSVT provides error processors with addressability to the common subroutines. The error processor link register gives addressability to the CSVT and so the first section of the DSECT overlays the code required to branch around the actual table.

```
DFHNEPC    DSECT
           DS      F          Load instruction
           DS      F          Branch instruction
CSVTNEP    DS      A          Node error program base address
CSVTESBL   DS      A          NEPESBL - ESB locate routine
CSVTNEBD   DS      A          NEPNEBD - NEB delete routine
CSVTECUP   DS      A          NEPECUP - error count update routine
```

User-written node error programs

You can write several node error programs, as described in "Multiple NEPs" on page 116. When an error occurs, the node abnormal condition program passes control to an interface module, DFHZNEPI, which determines the transaction class and passes control to the appropriate node error program.

If only one node error program is used, the interface module (DFHZNEPI) is not required. If the node error program is named DFHZNEP, the node abnormal condition program will branch directly to that. If more than one node error program is used, the interface module (DFHZNEPI) is required. In this case, the node error programs must be given names other than DFHZNEP. Every node error program generated must be defined in the processing program table (PPT) by means of a DFHPPT TYPE=ENTRY macro instruction, or in the CSD file.

DFHZNEPI macros

The following macros are required to generate the node error program interface module (DFHZNEPI):

- DFHZNEPI TYPE=INITIAL – to specify the name of the default transaction-class routine.
- DFHZNEPI TYPE=ENTRY – to associate the transaction-class with your transaction-class error handling routine.
- DFHZNEPI TYPE=FINAL – to end the DFHZNEPI macro instructions.

The DFHZNEPI interface module must be generated when you require the node abnormal condition program to pass control to the appropriate user-written node error program for resolution of the error.

Default transaction-class routine – DFHZNEPI TYPE=INITIAL

The DFHZNEPI TYPE=INITIAL macro instruction specifies the name of the default transaction-class routine to be used for the DFNZNEPI module.

```
DFHZNEPI  TYPE=INITIAL
          [,DEFAULT=name]
```

DEFAULT=name

Specifies the name of the default transaction-class routine to be used. A link will be made to this default routine under any one of three conditions:

- Specification of DFHPCT TYPE=ENTRY,NEPCLAS=0 (default).
- Specification of DFHPCT TYPE=ENTRY,NEPCLAS=value >255.
- No transaction-class routine has been specified via the DFHZNEPI TYPE=ENTRY macro for the transaction-class value identified by the DFHPCT TYPE=ENTRY,NEPCLAS=integer specification.

If any one of these conditions is true, and a DEFAULT=name operand has not been specified, then no routine is invoked.

The DFHZNEPI TYPE=INITIAL instruction must always be specified, and must be placed before any other forms of the DFHZNEPI macro instructions. Only one TYPE=INITIAL macro may be specified.

Transaction-class error-handling routine — DFHZNEPI TYPE=ENTRY

You use the DFHZNEPI TYPE=ENTRY macro instruction to associate the transaction-class, specified in the NEPCLAS=integer operand of the DFHPCT TYPE=ENTRY instruction, with your transaction-class error handling routine. The format of this macro instruction is as follows:

```
DFHZNEPI  TYPE=ENTRY
          ,NEPCLAS=integer
          ,NEPNAME=name
```

NEPCLAS=integer
Specifies the transaction-class, and must be in the range 1 through 255. No value should be specified that has been specified in a previous DFHZNEPI TYPE=ENTRY instruction.

NEPNAME=name
Specifies a name for the transaction-class routine to be associated with the specified transaction-class. An error condition will result if the name is specified either as DFHZNEP, or is greater than 8 characters.

Both the TYPE=ENTRY operands must be specified.

Terminate entries — DFHZNEPI TYPE=FINAL

```
DFHZNEPI  TYPE=FINAL
```

TYPE=FINAL
Completes the definition of module DFHZNEP and must be specified last. The assembly should be terminated by an END statement with no entry name specified, or by the statement: END DFHZNENA.

3270 unavailable printer

This condition arises when a print request is made using the 3270 print request facility, and there are no printers on the control unit, or when the printer(s) is (are) in one of the following conditions:

- Out of service
- Not in transceive or receive status for automatic transaction initiation
- A task is presently attached
- Busy on a previous operation
- Intervention required.

The procedure is applicable to 3270 logical units or the 3270 compatibility mode logical unit when using the PRINTTO and ALTPRT operands of the DFHTCT TYPE=TERMINAL macro.

The terminal control program recognizes this condition, and issues a READ BUFFER operation to collect the data into a terminal I/O area. The TIOA is of the same format as it is when an application program has issued a terminal control read buffer request.

The terminal control program VTAM section (DFHZCP) then queues the TCTTE to the node abnormal condition program with the error code X'42' (TCZCUNPRT). The fields relevant to this situation are:

TCTTEDA	Data address area
TWAPRNT	Field for node error program to return information to the node abnormal condition program. Set to 0 on initial entry to node error program.

The node abnormal condition program writes the DFH2497 'UNAVAILABLE PRINTER' or the DFH3493 'INVALID DEVICE TYPE FOR A PRINT REQUEST' message to the CSMT destination and links to the node error program with no default actions set.

On return from node error program, the node abnormal condition program will perform the following actions, based upon the TWAPRNT field in the TWA:

1. If TWAPRNT is all FFs (−1), the node abnormal condition program assumes that node error program has disposed of the data to be printed.
2. If TWAPRNT is zero, the node abnormal condition program assumes that no printer is available.
3. If TWAPRNT is neither −1 or zero, the node abnormal condition program assumes that TWAPRNT is the address of the printer. An interval control PUT will be performed to the provided terminal. The transaction to be initiated is CSPP (print program), and the time interval will be zero.
   a. If an error occurs on the interval control PUT, the node abnormal condition program will write the DFH2496 IC FAILURE message to the destination CSMT. The node abnormal condition program will then link to the node error program again with the TWAPRNT field set to −2. This is done in order for the node error program to have a last chance to dispose of the data. Upon the second return from node error program to the node abnormal condition program, the node abnormal condition program will reexamine TWAPRNT. If TWAPRNT is −1, this indicates that the node error program has disposed of the data.
   b. If no error occurred on the interval control PUT, the node abnormal condition program will check for the following printer conditions:

      Out of service
      Intervention required
      Any condition other than RECEIVE or TRANSCEIVE status.

If one of these conditions is true, the node abnormal condition program will issue the DFH2495 PRINTER OUTSERV/IR/INELIGIBLE-REQ QUEUED message to the destination CSMT.

4. The node abnormal condition program will then terminate any PRINT requests on the originating terminal and will perform normal action flag processing on that terminal.

Session failures

Following certain categories of error associated with logical unit or path failures, the session between CICS and the logical unit may be lost. The default action taken by DFHZNAC may be to put the TCTTE out of service.

A method of automatically reacquiring the session is for your node error program to alter the default DFHZNAC actions and to keep the TCTTE in service. The node error program can then issue an interval control PUT or INITIATE macro against that TCTTE (through the TRMIDNT) with a transaction written in a similar manner to the CICS good morning signon message (CSGM). When the transaction is initiated using automatic task initiation (ATI), CICS will try to reacquire the session. If the session fails again, DFHZNAC will be reinvoked and the process will be repeated.

The time specified in the interval control PUT or INITIATE macro instruction would be determined by installation-dependent expected-mean-time-to values for that installation.

If used in this way, the initiated transaction can write an appropriate signon message when the session has been acquired. Note, however, that if GMMSG=YES is specified in DFHTCT TYPE=TERMINAL, the CICS good morning message will also be initiated at session open time.

The node error program in an XRF environment

If you are using the extended recovery facility (XRF), a VTAM failure in your active CICS system may cause a takeover by the alternate CICS system. Each terminal from the failing system that is switched to the alternate system will be passed to DFHZNAC for 'conversation restart' processing. This is similar to 'session opened' processing for a normal session start.

One of the steps of the 'conversation restart' process is to LINK to the node error program with the error code X'3F'. You can write your own error processor to handle error code X'3F' if you want to be able to change any of the recovery notification options (the recovery notification, the recovery message and the recovery transaction) for some of the switched terminals. The following parameters are passed to the node error program in the transaction work area (TWA) of DFHZNAC.

```
TWAXRNOT DS   X             Recovery Notification Options
TWAXRNON EQU  X'80'         Recov Notification = None
TWAXRMSG EQU  X'40'         Recov Notification = Message
TWAXRTRN EQU  X'20'         Recov Notification = Transaction
*
TWAXMSTN DS   CL8           Recovery Mapset Name
TWAXMAPN DS   CL8           Recovery Map Name
*
TWAXTRAN DS   CL4           Recovery Transaction ID
```

Changing the recovery notification

The method of recovery notification for a terminal is defined using the RECOVNOTIFY option of the TYPETERM definition, which is described in the *CICS/MVS Resource Definition (Online)* manual. This is the most efficient way to specify the recovery notification method for the whole network, because CICS initiates the notification procedure during the early stages of takeover. However, you may want to change the recovery notification method for some of the switched terminals, and you can use the node error program to make this change. For example, you may want most terminals of a particular type to receive the recovery message at takeover, and the rest to get no notification that service has been restored. You would code RECOVNOTIFY(MESSAGE) in the TYPETERM definition, and then use the node error program to change the recovery notification to NONE for the few terminals that are not to be notified.

Changing the recovery message

If you define a terminal with RECOVNOTIFY(MESSAGE) in its TYPETERM, a recovery message will be sent to the terminal after takeover. There is a CICS-supplied message in BMS map DFHXREC of map set DFHXMSG. Its text is:

> CICS/MVS has recovered after a system failure
> Execute recovery procedures.

You can specify your own map set in the node error program if you wish to change the recovery message for some of the switched terminals. This could be useful, for example, if signon security is in force and you want to tell terminal users to sign on again. The map set that you specify must have a PPT entry. If you wish to change the recovery message for all terminals, it would be more efficient to replace the CICS-supplied map with your own.

Changing the recovery transaction

The recovery transaction that is to be started at a terminal after takeover is specified using the RMTRAN operand of the DFHSIT macro. This is the most efficient way of specifying a recovery transaction for the network. You can specify a different transaction for some of the switched terminals by overwriting the TWAXTRAN value in the transaction work area of DFHZNAC. The transaction that you specify must have a PCT entry, and the terminal must be defined with the option ATI(YES).

Chapter 2.11. The extended recovery facility overseer program

Note: The information in this chapter is of interest only to users of the extended recovery facility (XRF). An overview of XRF is provided in the *CICS/MVS XRF Guide*.

Introduction

The extended recovery facility overseer program has two major functions, which are:

- To display current status information of active and alternate CICS regions
- To restart failed CICS regions in place without operator intervention.

There is a CICS-supplied sample overseer program which performs these two functions and which you may find adequate for your installation. The sample program is described below, and the following are provided:

- A description of the main functions performed by the sample code
- A description of the sample program's interface to CICS
- Instructions for telling the overseer program which active and alternate pairs to monitor.

You can customize the sample program if you want to change parts of the code or to extend the range of functions performed by the overseer, and this process is described beginning on page 152.

The sample overseer program

The CICS-supplied sample overseer is an assembler-language batch program that runs in its own address space. The source of the sample program is in member DFH$AXRO of CICS212.SAMPLIB, and its associated DSECTs are supplied in member DFH$XRDS of the same library. An assembled version of DFH$AXRO is supplied in CICS212.LOADLIB. The program acts on four commands entered by the console operator. These are:

D to display the current status of all active and alternate pairs being monitored by the overseer program

R to enable or disable the restart-in-place function of the overseer program

S to take a snap dump of the sample program

E to terminate the sample program.

The full format of the operator command entered at the MVS console is:

```
MODIFY OVERSEER,command identifier
```

where 'command identifier' is D, R, S, or E. The 'D' command and the 'R' command control the two major functions of the sample overseer program (display and restart-in-place), and descriptions of these follow.

The display function

When the operator enters the 'D' command at the MVS console, the sample overseer program issues a multiline write-to-operator command (MLWTO) showing the last known state of each of the active and alternate pairs that it is monitoring. The overseer retrieves this information from the control and message data sets, in which the CICS availability manager (CAVM) has been recording state and surveillance information. The display includes a title line and one line of status information for each active and alternate pair. The title line is as follows:

```
GEN-APP  ACT-JOB  ACT-APP  ACPU A-ST  BKP-JOB  BKP-APP  BCPU B-ST
```

Each line of status information provides the following:

- The generic applid of the active and alternate pair (GEN-APP)
- The CICS jobname of the active (ACT-JOB) and of the alternate (BKP-JOB)
- The specific applid of the active (ACT-APP) and of the alternate (BKP-APP)
- The SMF IDs of the CPUs on which the active and the alternate were last known to be executing (ACPU and BCPU)
- The last known status of the active (A-ST) and of the alternate (B-ST).

The status value can be one of the following:

ACT	Active signed on normally and running the active CICS workload
BKP	Alternate signed on and running normally
SOFN	Signed off normally
SOFA	Signed off abnormally
TKOV	Taking over (alternate only)
INCA	Incipient active, meaning that an alternate CICS is taking over from an active CICS. The active job has signed off abnormally, and the incipient active is waiting for the active job to terminate.

An example of the status display is shown in the run-time example on page 144.

Note: An 'X' following any of these status values indicates that the associated job is currently executing. However, because JES services are used to discover the execution state of a job, only those jobs that are running on the same JES as the overseer program, or on another JES spool member, will have the correct execution state. Any job that is on a different JES shared spool will appear not to be executing.

The restart-in-place function

The overseer program can restart failed CICS regions in place automatically, provided that they are on the same CEC as the overseer. The alternatives to automatic restart are operator-initiated restart, automatic takeover to the alternate, and operator-initiated takeover.

The usefulness to you of automatic restart in place will depend on the configuration of your system. If you are operating a two-CEC single region system, or several independent regions, you can allow the overseer to restart an active region in place automatically when it fails, you can choose automatic takeover by the alternate, or you can leave the operator to decide what to do. The operator could decide to restart the failing region in place or to initiate a takeover by the alternate, and this decision is likely to depend on which part of your system has failed.

If you are operating a single-CEC MRO system, the failure of an active region can be handled by a takeover by the alternate, without causing all the related regions to be taken over, because the new active region can continue communication with the other active regions. Takeover is therefore likely to be your preferred course of action.

Automatic restart in place of failed regions is most useful in the two-CEC MRO environment. Because related regions must operate on the same CEC, a takeover of one region means that all related regions must also be taken over by their alternates. A region may not be important enough for you to want every failure to cause a takeover to the alternate CEC. This could disrupt users who would not otherwise have been affected by the failure. Automatic restart in place of the failed region is therefore likely to be preferred to takeover in these circumstances.

Enabling and disabling restart in place

The restart-in-place function of the overseer program can be enabled and disabled using the 'R' command. When you enter this command, restart processing is enabled or disabled for all generic applids that the overseer is monitoring. You can also specify that particular active and alternate pairs are not to be automatically restarted in place, regardless of whether restart processing is enabled or disabled. This is described in "How to tell the sample overseer which actives and alternates to monitor" on page 143.

The 'R' command works like an ON/OFF switch. Restart in place is enabled when the sample program is initialized. When the 'R' command is first entered, restart in place is disabled. If you issue the command again, restart will be enabled again, and so on. If a region fails while restart in place is disabled, no attempt to restart it will be made, even if restart in place is enabled again.

Rules that control restart in place

The sample overseer program concludes that a region has failed if

1. the region is not executing now, and was known to have been executing during the previous examination of the relevant CAVM data sets by the overseer and,
2. the region did not sign off normally from the CICS availability manager (CAVM).

The overseer program can restart a failed **active** region in place, provided that the following conditions are met:

- Restart in place is enabled for this overseer.
- Restart in place is enabled for this active and alternate pair.
- There is no corresponding executing alternate region, or the alternate region is currently defined with TAKEOVER=COMMAND. If the alternate region is defined with TAKEOVER=AUTO or TAKEOVER=MANUAL, the overseer assumes that the alternate will initiate a takeover or that the console operator will decide what action to take.
- The failing region was running on the same CEC as the overseer.
- An attempt to restart the region in place is not already in progress.
- If the failing region belongs to a group of related regions (an MRO environment, for example), a takeover to another CEC, perhaps initiated by another region, is not under way.

When a failed active region is restarted in place, whether by the operator or by the overseer, the corresponding alternate region cannot continue to support the new active region, and must be restarted. The overseer program will restart the **alternate** region automatically in these circumstances, provided that restart processing is enabled for both the failing region and the overseer. This rule applies generally to the restarting of failed regions.

If you want to be able to restart regions in place on both CECs in a two-CEC environment, an overseer program must execute on each CEC.

If the failed region was started originally as a started task, the overseer program restarts it as a started task, and if the failed region was started as a job, the overseer restarts it as a job. For more information about how the sample overseer program restarts failed regions in place, see the *CICS/MVS Operations Guide*.

How the sample overseer program interfaces with CICS

The overseer service is made up of a CICS overseer module (name DFHWOS), which you cannot customize, and a CICS-supplied sample overseer program (module name DFH$AXRO), which you can customize or replace with your own overseer program. DFHWOS loads the overseer program. DFHWOS and DFH$AXRO are supplied in CICS212.LOADLIB.

The CICS overseer module DFHWOS provides a stable interface to the CAVM datasets and to certain MVS-authorized services that the overseer program requires. The overseer program invokes those services by means of a CICS-supplied group of macros called the DFHWOSM macros, which are described beginning on page 145.

DFHWOS therefore invokes the sample program, and is subsequently invoked by the sample program whenever the sample issues a DFHWOSM macro. The DFHWOSM macros do not interact directly with either the active or the alternate CICS address spaces.

How to tell the sample overseer which actives and alternates to monitor

The sample overseer program is written to handle 20 active and alternate pairs and 20 related system names. You can increase or decrease these numbers by changing the values of the variables GENSIZE (active and alternate pairs) and RLTSIZE (related system names) in DFH$AXRO. The maximum number of active and alternate pairs that can be monitored by the overseer is 50.

A 'related system name' identifies those regions or systems that cannot be considered in isolation by the overseer. The most common example of this is an MRO environment, where the overseer needs to be able to identify related regions when deciding whether to restart a failed region in place. Those regions or systems that are identified with a common related system name must execute on the same CEC.

The sample program discovers which active and alternate pairs it is monitoring from a VSAM key-sequenced data set called DFHOSD, which contains a single entry for each active and alternate pair. You create this data set and initialize it with information about active and alternate pairs before you use the overseer for the first time. You will also have to redefine the DFHOSD data set whenever you want to change the information that it holds. A sample job stream is provided which you can use to:

- Delete and define the DFHOSD data set
- Initialize the DFHOSD data set with information about sample active and alternate pairs
- Execute the overseer code and the sample overseer program.

The sample job stream is called DFHIVXRO and is in CICS220.INSTLIB. It is described in the *CICS/MVS Operations Guide*.

The sample overseer program reads the DFHOSD records in key sequence and builds a table of entries, stopping when the first 20 entries have been read.

Each active and alternate pair is known by its generic applid on this data set. Every entry on the data set contains the following information:

- A 12-byte key field, containing the 4-byte value 'GNbb' followed by the 8-byte generic applid of the active and alternate pair.
- The DDNAMEs of the Control Data Set and the Message Data Set associated with this generic applid. Each of these is an 8-byte value.
- An optional 8-byte RELATEID, to identify related systems.
- A restart-in-place indicator to show whether a region can be restarted in place. The only value that will prevent an attempt to restart in place is 'N'.

The data structure of the DFHOSD data set entries is provided in member DFH$XRDS of CICS212.SAMPLIB.

A run-time example

During its execution, the overseer program keeps the operator informed of its operations and of any errors by issuing messages to the MVS console. The first line of every message from the overseer begins with the characters +OVERSEER:. A complete list of these messages, with an explanation of each, is provided in the prolog of the source code listing of the sample overseer program. The jobname (in this example, JOB 6840) uniquely identifies the overseer. To give you some idea of the overseer's output, here is a run-time example.

```
 18.00.49 JOB 6840  IEF403I OVERSEER - STARTED - TIME=18.00.49
 18.00.49 JOB 6840  +OVERSEER: ENTERING INITIALIZATION
 18.00.50 JOB 6840  +OVERSEER: DFHOSD DATASET OPENED

 18.01.12 JOB 6840  +OVERSEER: DFHOSD DATASET CLOSED
 18.01.12 JOB 6840  +OVERSEER: INITIALIZATION COMPLETE
 MODIFY OVERSEER,R
 18.01.29 JOB 6840  +OVERSEER: RESTART IN PLACE IS NOW DISABLED
 MODIFY OVERSEER,R
 18.01.36 JOB 6840  +OVERSEER: RESTART IN PLACE IS NOW ENABLED
 MODIFY OVERSEER,D
 18.01.41 JOB 6840  +OVERSEER: CPU CEC5 DISPLAY
                    +GEN-APP  ACT-JOB  ACT-APP  ACPU A-ST  BKP-JOB  BKP-APP
                     BCPU B-ST DBDCCICS DBCICSJ1 DBDCCIC1 CEC5 ACT X DBCICSJ2
                     DBDCCIC2 CEC4 SOFN
 MODIFY OVERSEER,S
 18.02.19 JOB 6840  $HASP375 OVERSEER ESTIMATED  LINES EXCEEDED
 18.02.23 JOB 6840  +OVERSEER: SNAP DUMP TAKEN
```

Job DBCICSJ1 fails and no alternate is available:

```
 18.03.04 JOB 6840  +OVERSEER: JOB IS BEING RESTARTED
 18.03.04 JOB 6840  START DFHCRST,CJOB=DBCICSJ1,CSTART=AUTO  (BY IOP)¹
 18.03.04 JOB 6840  START DFHCRST,CJOB=DBCICSJ1,CSTART=AUTO
 MODIFY OVERSEER,E
 18.03.38 JOB 6840  +OVERSEER: TERMINATION COMPLETE
 18.03.39 JOB 6840  IEF404I OVERSEER - ENDED - TIME=18.03.39
```

Figure 10. The overseer run-time example

[1] If the job were being resubmitted as a started task, rather than as a job, the restart messages would be as follows:

```
 18.03.04 STC 6840  START DBCICSJ1,CSTART=AUTO  (BY IOP)
 18.03.04 STC 6840  START DBCICSJ1,CSTART=AUTO
```

The message (BY IOP) tells you that restart has been initiated by the Installation Overseer Program (IOP).

The DFHWOSM macros

The DFHWOSM macros invoke the CICS module DFHWOS to provide services to the overseer program. The macros are the supported interface to the CAVM data sets, and are supplied to perform the following functions:

- Initialize access to the CAVM data sets for a named generic applid (DFHWOSM FUNC=OPEN)
- Terminate access to the CAVM data sets for a named generic applid (DFHWOSM FUNC=CLOSE)
- Retrieve status information for a named generic applid from the CAVM data sets (DFHWOSM FUNC=READ)
- Issue MVS commands (DFHWOSM FUNC=OSCMD)
- Discover current JES JOB status (DFHWOSM FUNC=JJS|QJJS)
- Issue a JES cancel for a named job (DFHWOSM FUNC=JJC).

Two additional macros (DFHWOSM FUNC=BUILD and DFHWOSM FUNC=TERM) are supplied to open and close communication with DFHWOS. The macros are described in detail beginning on page 146. For all the DFHWOSM macros, the following rules apply:

- The 'label' field is optional.
- If the macro has an input parameter list, the address of that parameter list must be supplied as the value of the PARM option. The address itself may be specified as a register number or as a label. Register 1 is the default value.
- If the macro has to supply either a BUILD TOKEN or an OPEN TOKEN to DFHWOS (as described in "The DFHWOSM tokens" below), the token must be provided in the register specified in the TOKEN option. Register 14 is the default register.

The DFHWOSM tokens

When DFHWOS first invokes the overseer program it passes a value in register 1 which is known as the **ENTRY** token. The ENTRY token value is stored by the overseer program on entry and is passed back to DFHWOS as input to the BUILD, OSCMD, JJS and JJC macros.

The DFHWOSM FUNC=BUILD macro must be the first macro issued by the overseer program and must complete successfully. The register 1 output from this macro is a second token called the **BUILD** token. The BUILD token value is stored by the overseer program and passed back to DFHWOS as input to the OPEN, CLOSE, READ, QJJS and TERM macros.

DFHWOSM FUNC=BUILD macro

This macro must be issued by the overseer program to initialize its communication with DFHWOS. No other macro can be issued by the overseer program until `DFHWOS FUNC=BUILD` has completed successfully.

```
label      DFHWOSM FUNC=BUILD
                   [,TOKEN={token register|14}]
```

Input

The TOKEN value is the ENTRY token that was passed to the sample overseer program when it was first invoked by DFHWOS.

Output

Register 1	Contains the BUILD token value, which must be returned as an input value by the overseer program on certain subsequent requests. This value will be returned to register 1 only if register 15 has a return code of 0.	
Register 15	Contains one of the following completion codes:	
	0	Communication successfully initialized between the overseer program and DFHWOS
	4	Incorrect TOKEN value supplied
	8	Insufficient storage.

DFHWOSM FUNC=OPEN macro

The `DFHWOSM FUNC=OPEN` macro initializes access to the CAVM data sets for a named generic applid.

```
label      DFHWOSM FUNC=OPEN
                   [,PARM={parm address|1}]
                   [,TOKEN={token register|14}]
```

Input

The PARM value is a pointer to three further addresses, and these are:

1. the address of the generic applid
2. the address of the ddname of the control data set
3. the address of the ddname of the message data set.

The TOKEN value is the BUILD token.

Output

Register 15	Contains one of the following completion codes:	
	0	Access initialized, active and alternate signed on
	1	Access initialized, active signed on

2	Access initialized, alternate signed on
3	Access initialized, nothing signed on
4	Same SMF MVS name; IPL time of active earlier than MVS IPL time
5	Same SMF MVS name; IPL time of alternate earlier than MVS IPL time
6	Insufficient storage
7	Generic applid is not associated with the named CAVM data sets
8	Access already initialized for this generic applid or for this ddname
C	Data set open failure
10	SHOWCB failure.

A register 15 return code value of 0 — 5 indicates that a DFHWOSM FUNC=READ macro can now be issued. A return code value of 6 or above indicates that the OPEN has failed and that the overseer program will not be able to access the CAVM data sets.

DFHWOSM FUNC=CLOSE macro

This macro terminates access to the CAVM data sets for a named generic applid.

```
label      DFHWOSM FUNC=CLOSE
                 [,PARM={parm address|1}]
                 [,TOKEN={token register|14}]
```

Input

The PARM value is a pointer to the address of the generic applid whose associated CAVM data sets are no longer to be accessed by the overseer program.

The TOKEN value is the BUILD token.

Output

Register 15 Completion codes:

0	CLOSE request was successful and the CAVM data sets associated with this generic applid can no longer be accessed by the overseer program
4	Incorrect TOKEN value supplied
8	Access to CAVM data sets for the named generic applid had not been initialized.

DFHWOSM FUNC=READ macro

This macro returns information about a named generic applid from its associated CAVM data sets.

```
label      DFHWOSM FUNC=READ
                 [,PARM={parm address|1}]
                 [,TOKEN={token register|14}]
```

Input

The PARM value is a pointer to a parameter list that contains the addresses of the generic applid and the dbllist. The dbllist is a list of one or more doublewords.

In the first two bytes of the second word of each of these doublewords you supply the 'DBLID' of the information you require. Each piece of information that you can request is identified by a DBLID, and a list of these is provided in Figure 12 on page 149.

The first word of each doubleword is an output area to contain the address of the requested information, and the last two bytes of the second word of each doubleword will contain the length of the information. The end of the dbllist is signalled by setting the high order bit of the last doubleword to one. Figure 11 illustrates the input to the READ macro.

The TOKEN value is the BUILD token.

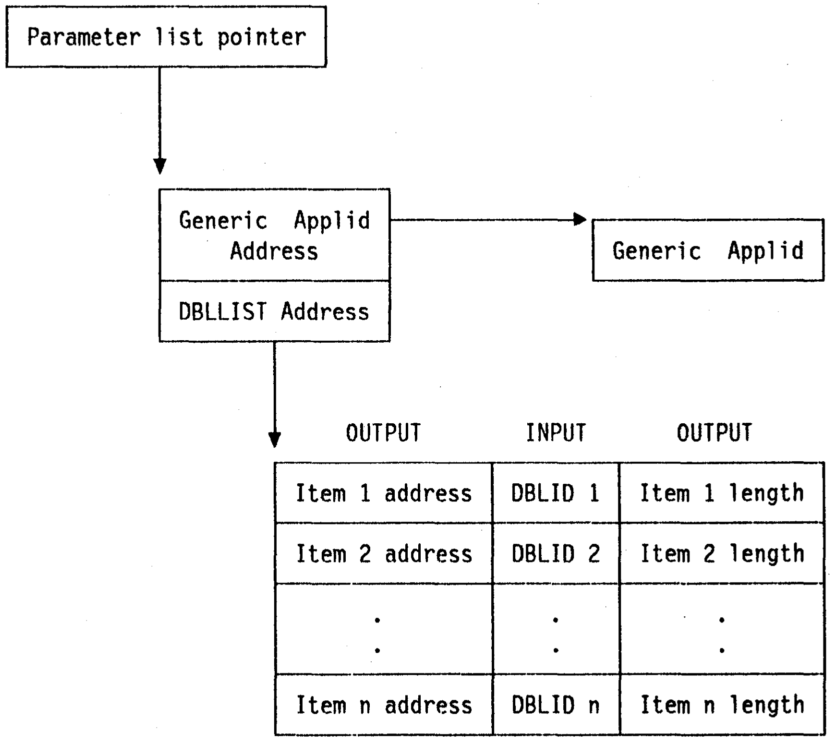


Figure 11. Input to the DFHWOSM FUNC=READ macro

```
DBLIDs for the active:

DBLID1   EQU   X'0001'              JOBNAME
DBLID2   EQU   X'0002'              JES JOBID
DBLID3   EQU   X'0003'              JOB SUBMISSION TIME (STIME)
DBLID4   EQU   X'0004'              JOB STEP TASK ATTACH TIME (ATIME)
DBLID5   EQU   X'0005'              CANCEL NAME
DBLID6   EQU   X'0006'              JES SSNAME
DBLID7   EQU   X'0007'              MVS SMF NAME
DBLID8   EQU   X'0008'              MVS IPL TIME
DBLID9   EQU   X'0009'              SPECIFIC APPL NAME
DBLID10  EQU   X'000A'              ADDRESS SPACE IDENTIFIER (ASID)
DBLID11  EQU   X'000B' TO X'001F' SPARE FOR STATE CTL ITEMS.
DBLID32  EQU   X'0020'              HEARTBEAT INTERVAL
DBLID33  EQU   X'0021'              HEARTBEAT COUNTER
DBLID34  EQU   X'0022'              MSG FILE CURSOR
DBLID35  EQU   X'0023'              STATUS VALUE (STATE)
DBLID36  EQU   X'0024'              INQUIRE HEALTHDATA

DBLIDs for the alternate:

DBLID257 EQU   X'0101'              JOBNAME
DBLID258 EQU   X'0102'              JES JOBID
DBLID259 EQU   X'0103'              JOB SUBMISSION TIME (STIME)
DBLID260 EQU   X'0104'              JOB STEP TASK ATTACH TIME (ATIME)
DBLID261 EQU   X'0105'              CANCEL NAME
DBLID262 EQU   X'0106'              JES SSNAME
DBLID263 EQU   X'0107'              MVS SMF NAME
DBLID264 EQU   X'0108'              MVS IPL TIME
DBLID265 EQU   X'0109'              SPECIFIC APPL NAME
DBLID266 EQU   X'010A'              ADDRESS SPACE IDENTIFIER (ASID)
DBLID267 EQU   X'010B' TO X'011F' SPARE FOR STATE CTL ITEMS
DBLID288 EQU   X'0120'              HEARTBEAT INTERVAL
DBLID289 EQU   X'0121'              HEARTBEAT COUNTER
DBLID290 EQU   X'0122'              MSG FILE CURSOR
DBLID291 EQU   X'0123'              STATUS VALUE (STATE)
DBLID292 EQU   X'0124'              INQUIRE HEALTHDATA
```

Figure 12. DBLIDs for the DFHWOSM FUNC=READ macro

Note: The data structures of the status information pointed to by items X'0024' and X'0124' are provided in DSECT DFHXRHDS of CICS212.MACLIB.

Output

Register 15 Contains one of the following completion codes:

0	Read successful, active and alternate signed on
1	Read successful, active signed on
2	Read successful, alternate signed on
3	Read successful, nothing signed on
4	Same SMF MVS name; IPL time of active earlier than MVS IPL time

5	Same SMF MVS name; IPL time of alternate earlier than MVS IPL time
8	CAVM data set access not initialized
10C	DBLID not known
1xx	Read subtask problem.

If a completion code of 0 — 5 is returned to register 15, each doubleword of the DBLLIST will contain the address (4 bytes) and the length (2 bytes) of the output from this read. A completion code of 8, 10C or 1xx indicates a READ failure.

DFHWOSM FUNC=TERM macro

This macro terminates communication between the overseer program and DFHWOS, and releases any associated storage. It must be issued before the overseer program completes to ensure an orderly termination.

```
label      DFHWOSM FUNC=TERM
                 [,TOKEN={token register|14}]
```

Input

The TOKEN value is the BUILD token.

Output

Register 15 Contains the following completion codes:

0	Communication terminated successfully
Nonzero	Request failed.

DFHWOSM FUNC=OSCMD macro

The OSCMD macro is used to issue MVS commands. (The overseer program performs restart in place of a failed region by issuing an OSCMD macro.) The text of the required MVS command is provided as input to the macro, and the OSCMD service issues an SVC 34 specifying this command text. In addition, the OSCMD service issues an MVS WTO request so that a copy of the command text appears on the MVS console to keep the operator informed of what is about to happen. This copy has the comment '(BY IOP)' appended to show that the command is going to be issued by an overseer program. A second copy of the command text is sent to the console when the MVS command is issued. The run-time example on page 144 includes an example of this.

```
label      DFHWOSM FUNC=OSCMD
                 [,PARM={parm address|1}]
                 [,TOKEN={token register|14}]
```

Input

The PARM value is a single address that points to a 'command area'. The command area is made up of a 4-byte length field followed by the command data. The length field contains the length of the whole command area. The command data must be in WTO command format.

The TOKEN value is the ENTRY token.

Output

Register 0	Completion code set by SVC 34 as a response to the MVS command that was issued by the OSCMD macro.
Register 15	Response to the OSCMD itself. A return code of 16 indicates that the OSCMD has failed.

DFHWOSM FUNC=[JJS|QJJS] macro

Given a JOBNAME and JES job identifier, both versions of this macro return the current JES job status into a copy of the JES subsystem options block (SSOB).

The FUNC=JJS macro returns control when the JES call has completed successfully or unsuccessfully. The FUNC=QJJS macro returns control immediately and posts an event control block (ECB) once the JES request has completed.

```
label      DFHWOSM FUNC=[JJS|QJJS]
                 [,PARM={parm address|1}]
                 [,TOKEN={token register|14}]
```

Input

For FUNC=JJS, the PARM value is a pointer to the addresses of the following:

An 8-byte jobname
An 8-byte JES job id
A 256-byte SSOB return area.

The TOKEN value is the ENTRY token.

For FUNC=QJJS, the PARM value is a pointer to the addresses of the following:

An 8-byte jobname
An 8-byte JES job id
A 256-byte SSOB return area
A doubleword area to hold two ECBs.

The QJJS macro requires 2 ECBs: the first is posted when the JES call completes, the second is posted if a timeout occurs before JES returns.

The TOKEN value is the BUILD token.

Output

Register 15 Contains the following completion codes:

0	JES status returned as requested in the SSOB return area
Nonzero	Return code from JES.

DFHWOSM FUNC=JJC macro

This macro issues a JES cancel for a named job with a JES job identifier. Note that this macro will not cancel a started task.

```
        DFHWOSM FUNC=JJC
              [,PARM={parm address|1}]
              [,TOKEN={token register|14}]
```

Input

The PARM value is a pointer to the addresses of the following:

An 8-byte jobname
An 8-byte JES job id
A 256-byte SSOB return area.

The TOKEN value is the ENTRY token.

Output

Register 15	Contains the following completion codes:	
	0	JES cancel completed. SSOB and status array returned from JES.
	Nonzero	Return code from JES.

DFHWOSM FUNC=DSECT macro

This macro generates a number of DSECTs, the most useful of which is the DSECT of the DBLID definitions.

```
DFHWOSM FUNC=DSECT
```

Customizing the sample program

The sample overseer program is written as a main body of code with two main subroutines, each written as a separate CSECT. One of these subroutines is called to perform the display of status information (CSECT DFH$ADSP), and one is called to perform restart in place (CSECT DFH$ARES). The associated DSECTs are provided in member DFH$XRDS of CICS212.SAMPLIB. There are a number of ways in which you can change the supplied code if you want to make the overseer program more suitable for your installation.

If you do change the overseer code in any way, you should note that the libraries SYS1.MACLIB and SYS1.AMODGEN are required for the assembly, and that the link-edit jobstep requires an entry name of the form ENTRY DFHXRONA. The supplied sample can be linked with the reentrant option (RENT).

Here are some customization suggestions:

- If the supplied display of status information (DSECT DSPDS) is not suitable, you can change the layout for your installation.
- The CSECT DFH$ADSP can be customized so that, for example, status information is displayed automatically at regular intervals, or whenever a region is in trouble, as well as when the console operator enters the 'D' command. This would require interpretation of the status information by the overseer.
- Any of the messages to the system console, which are listed in the prolog of the source module DFH$AXRO, can be changed.
- You can change the format or the content of the DFHOSD data set (DSECT OSDDS) if, for example, you want it to contain more information.
- You can change the restart function so that, for example, a failed region will be restarted only during periods of heavy use, while at other times a takeover to the alternate will be initiated by the operator.
- When an active region fails and is taken over to the alternate, the old active region must be restarted as the new alternate. In those cases where the cause of the takeover was not a CEC failure, restart of the old active as an alternate region could be automated in the overseer program.
- If you want to extend the function of the overseer program, you can incorporate the CEBT command, which is normally issued by the console operator to control the alternate. The CEBT command is described in the *CICS/MVS CICS-Supplied Transactions* manual.

  All of the CEBT functions are available for use in the overseer program, though it is unlikely that you will find it helpful to automate all of them, and there would, in some cases, be difficulties in handling the responses from the INQUIRE-type commands. However, it can be helpful for you to be able to automate the takeover process in some circumstances. Here are two examples of situations in which you could use the CEBT command to influence or to initiate takeover from the overseer program.

  1. The active CICS may place error information in the CAVM data sets when a VTAM failure occurs, depending on whether you have coded an exit program at the global user exit point XRRSTAT, and how you have coded it. (An exit program at this point can be used to decide whether or not VTAM failure data is recorded in the CAVM data sets.) If such data is placed in the CAVM data sets, information about the last eight failures detected by the active CICS region is available to the overseer. The overseer can evaluate this information and, if necessary, initiate a takeover by issuing the following CEBT command:

     ```
     MODIFY nnnn,CEBT PERFORM TAKEOVER
     ```

     where nnnn is a region name. In this case, you should ensure that the actions taken by the global user exit program at exit point XXRSTAT do not conflict with or duplicate those taken by the overseer program. For example, it would be possible for the global user exit program to request a CICS abend, and thereby initiate a takeover, and for the overseer program to issue the PERFORM TAKEOVER command while acting on the same information.

2. At certain times of the day, perhaps when fewer operational staff are available than at other times, you may find it convenient to change the TAKEOVER setting for some, or all, of your regions. For example, you can change the TAKEOVER value for a region from COMMAND or MANUAL to AUTO, without shutting down the alternate, so that takeover will be automatic until the setting is next changed. The CEBT command is as follows:

   ```
   MODIFY region_name, CEBT SET TAKEOVER AUTO
   ```

   In both of these examples, you would include takeover commands in the command list tables (CLTs) of these regions to ensure that their related regions are also switched when appropriate.

There is one optional section of code in the overseer program, which is described below.

Loop or wait detection

The sample overseer program includes some code that you can use to detect possible loops or waits in the active CICS region. The sample program monitors the CICS TCB (task control block) time stamp. If this remains the same for a period defined by the variable LOOPTM, a message is sent to the console warning of a possible loop or wait. The value of LOOPTM is the number of seconds (wait time) before a loop is suspected, and may need to be changed to suit your requirements and to avoid the detection of 'false' loops. It should be set to a value greater than the largest runaway task time interval (as specified on the DFHSIT operand ICVR) to avoid detection of user transaction loops. To include this LOOP WARNING code, set the variable &LOOPWARN to '1' and reassemble the sample.

Part 3. Journal management

Chapter 3.1. Use of specialized journal functions

A journal consists of a series of tape volumes or disk data sets used sequentially. Each tape volume comprises one data set.

This chapter contains information on the use of specialized journal functions with particular reference to user journals. The following topics are discussed:

- Customization programming within the journaling process; for example, opening, closing and reading journals.
- The layout and contents of journal records.
- Methods of reading journal data sets. The options are:
  - Offline:
    - Forward
    - Backward.
  - During execution:
    - Forward
    - Backward.
- User replaceable modules.

The information in this chapter should be read in conjunction with the discussion on journaling in the *CICS/MVS Application Programmer's Reference* manual and *CICS/VS Application Programmer's Reference Manual (Macro Level).* An overview of journal records is provided in the *CICS/MVS Recovery and Restart Guide.*

Customization programming

Journal records are written either directly from a user application program using the journal control commands or macros, or from a CICS management program on behalf of a user application. Processing these journal records is the user's responsibility. Typically, the system programmer writes the programs that open, close and read the journal data sets.

This section describes the following variants of the journal control macro instruction:

- DFHJC TYPE=OPEN, which opens a data set.
- DFHJC TYPE=CLOSE, which closes a journal data set.
- DFHJC TYPE=GETB or TYPE=GETF, which reads records backward or forward from a journal data set.

The program that issues these macro instructions must include a COPY DFHJCADS statement and the symbol JCABAR as a base register, to define and address the journal control area (JCA).

The JCA and the DFHJC macro instructions used to place records in a journal data set are described in the *CICS/VS Application Programmer's Reference Manual (Macro Level)*. The JOURNAL command is described in the *CICS/MVS Application Programmer's Reference* manual.

Opening a journal volume or data set – DFHJC TYPE=OPEN

The general format of the DFHJC macro instruction to open journal data sets is as follows:

```
DFHJC   TYPE=(OPEN[,{INPUT|OUTPUT}])
        [,IDERROR=symbolic-address]
        [,INVREQ=symbolic-address]
        [,IOERROR=symbolic-address]
        [,JFILEID={SYSTEM|nn|YES}]
        [,NORESP=symbolic-address]
        [,SIVOL=YES]
        [,STATERR=symbolic-address]
        [,VOLERR=symbolic-address]
        [,VOLUME={NEXT|PREVIOUS|CURRENT|FIRST}]
```

TYPE=(OPEN[,{INPUT|OUTPUT}])
Indicates that a data set of the specified journal is to be opened.

OPEN,INPUT
Indicates that the journal volume is to be opened for input. Before the volume is opened for input, the task must have obtained exclusive control of the journal by previously closing it.

OPEN,OUTPUT
Indicates that the journal volume is to be opened for output. Exclusive control of the journal, if obtained by a previous close operation, is relinquished.

OPEN
Is used as the first request for OPEN=DEFERRED journals. See the *CICS/MVS Recovery and Restart Guide* for further details.

If a journal is defined as OPEN=DEFERRED in DFHJCT TYPE=ENTRY, it must first be opened for output. Before issuing any open for input, the file is closed to get exclusive control.

IDERROR=symbolic-address
Specifies the address to which control is passed if the specified journal does not exist in the journal control table (JCT).

INVREQ=symbolic-address
Specifies the address to which control is passed if the TYPE of request is invalid. Note that journals to be opened for input must be specified with JOUROPT=INPUT in the JCT.

IOERROR=symbolic-address
Specifies the address to which control is passed if the operating system open fails.

JFILEID={SYSTEM|nn|YES}
Identifies the journal to be opened. The default is JFILEID=SYSTEM.

SYSTEM
Indicates that the journal is the system log.

nn Is a decimal value from 2 through 99, which identifies the journal.

YES
Indicates that the journal identification has been previously loaded in the journal control area field JCAJFID.

NORESP=symbolic-address
Specifies the address to which control is passed if the requested operation is successful.

SIVOL=YES **Tape journals only**
Indicates, for TYPE=(OPEN,INPUT) requests, that a specific volume is required. The VOLUME keyword must also be present to specify positioning; however, VOLUME=CURRENT is invalid because SIVOL identifies a specific volume.

Note: SIVOL=YES is an invalid request for disk journals because all CICS disk journal data sets must be permanently mounted.

| You must identify the tape being used. For unlabeled tapes, use JCARST, JCAVCD, and JCAVSN. For labeled tapes, use JCAVOLID, JCAFLG, and JCAPRTNO. For full details, refer to the section on reading journal data sets in the *CICS/MVS Operations Guide*.

The data to be placed in these fields must be obtained in advance by issuing a DFHJC TYPE=NOTE request. See "Reading journal data sets — DFHJC TYPE=GET" on page 162.

STATERR=symbolic-address
Specifies the address to which control is passed if the current status of the journal prevents the requested operation. For example, the request is to OPEN a journal that is already open. A status error code is also returned if the request attempts to open a journal already under exclusive control of a different task.

VOLERR=symbolic-address
Specifies the address to which control is passed if a volume error occurs. Possible errors include:

- Volume does not exist
- Volume cannot be located
- Volume no longer contains the required part of the journal.

VOLUME={NEXT|PREVIOUS|CURRENT|FIRST}
Specifies which volume of the journal is required, and how that volume is to be positioned when opened. The default is VOLUME=NEXT. NEXT and PREVIOUS refer to the time sequence in which the volumes are written. For journals using standard-labeled tape volumes, journal control will request that the appropriate volume is mounted. For journals using unlabeled tape volumes, it is the operator's responsibility to ensure that tape journal volumes are kept and mounted in sequence. Disk journal volumes are

permanently mounted, and journal control performs any necessary volume switching or positioning.

NEXT
Indicates, for TYPE=(OPEN,OUTPUT) requests, that journal output is to be continued from the start of the next reel or data set. For tape, a reel that is ready to be overwritten must be mounted.

For disk, a data set will be reused. However, if the journal was previously in input mode, VOLUME=NEXT will be ignored and VOLUME=CURRENT will be forced.

For TYPE=(OPEN,INPUT) requests, the next volume in chronological sequence is to be mounted, if necessary, opened for input, and positioned at the start of the data set.

PREVIOUS
Indicates, for TYPE=(OPEN,INPUT) requests, that the previous volume in chronological sequence is to be mounted, if necessary, opened for input and positioned at the end of the data set.

CURRENT
Indicates, for TYPE=(OPEN,INPUT) requests, that the current output volume (that is, the tape reel or disk data set that most recently received output) is to be opened for input and positioned at the end of data on the volume.

Note: If the current tape output reel was closed with LEAVE=YES, no remounting or repositioning delay should occur.

For TYPE=(OPEN,OUTPUT) requests, the current output volume is to be opened for output. For tape journals, this request is treated the same as VOLUME=NEXT, that is, a new output volume is begun. For disk, the journal is repositioned so that output continues after the last record previously written.

FIRST
Can only be used if OPEN=DEFERRED is specified in DFHJCT TYPE=ENTRY. VOLUME=FIRST has the same effect as VOLUME=CURRENT, except that the sequence number for this first volume of the run is initialized at 001. VOLUME=FIRST is only supported for output, except for the system log in a CICS with DL/I environment.

Note: During system initialization, all journals included in the journal control table are opened with TYPE=(OPEN,OUTPUT), VOLUME=FIRST (unless OPEN=DEFERRED was specified).

Note: VOLERR, STATERR, IDERROR, INVREQ, IOERROR and NORESP may be specified in a separate DFHJC TYPE=CHECK macro.

Closing a journal data set — DFHJC TYPE=CLOSE

The general format of the DFHJC macro instruction used to close a journal data set is described below.

```
DFHJC   TYPE=CLOSE
        [,IDERROR=symbolic-address]
        [,IOERROR=symbolic-address]
        [,JFILEID={SYSTEM|nn|YES}]
        [,LEAVE={NO|YES}]
        [,NORESP=symbolic-address]
        [,STATERR=symbolic-address]
```

TYPE=CLOSE
Indicates that the open volume in the specified journal is to be closed. Exclusive control of the journal is given to the requesting task. Hence, only the task that has issued the close request can reopen the journal.

To avoid tasks waiting on journal control writes and the possibility of a deadlock occurring, monitoring should be terminated before the journal is closed.

IDERROR=symbolic-address
Specifies the address to which control is to be passed if an entry for the specified journal does not exist in the journal control table.

IOERROR=symbolic address
Specifies the address to which control is to be passed if an I/O error occurs.

JFILEID={SYSTEM|nn|YES}
Identifies the journal to be closed. The default is JFILEID=SYSTEM.

SYSTEM
Indicates that the journal is the system log.

nn Is a decimal value from 2 through 99, which identifies the journal.

YES
Indicates that the journal identification has been previously loaded in the journal control area field JCAJFID.

LEAVE={NO|YES}
Indicates the positioning for tape journal files. The default is LEAVE=NO. The LEAVE keyword is ignored for disk files.

NO
Indicates that the reel is to be rewound and unloaded.

YES
Indicates that the reel is to remain ready and mounted, positioned at the end of the file.

NORESP=symbolic-address
Specifies the address to which control is to be passed if the requested operation is successful.

STATERR=symbolic-address
Specifies the address to which control is passed if the current status of the journal prevents the requested operation; for example, if the request is to CLOSE a journal which includes no open volumes. A status error code is also returned if the request attempts to close a journal already under exclusive control of a different task.

Reading journal data sets – DFHJC TYPE=GET

The general format of the DFHJC macro instruction used for finding the position in and reading from journal records is as follows:

```
DFHJC   TYPE={GETB|GETF|NOTE|POINT}
        [,EOFADDR=symbolic-address]
        [,IDERROR=symbolic-address]
        [,INVREQ=symbolic-address]
        [,IOERROR=symbolic-address]
        [,JFILEID={SYSTEM|nn|YES}]
        [,NORESP=symbolic-address]
        [,NOTOPEN=symbolic-address]
        [,STATERR=symbolic-address]
        [,VOLERR=symbolic-address]
```

The system acquires an input area into which the journal record is moved. The address of this area is returned in the field JCAADATA. You must use this address to establish addressability to the area, which is defined by the DFHJCRDS DSECT. See "Layout and contents of journal records" on page 164.

TYPE={GETB|GETF|NOTE|POINT}
Indicates the journal operation required.

GETB
Retrieves the journal record preceding the current position.

GETF
Retrieves the next journal record.

For TYPE=GETB and TYPE=GETF requests, the address of the journal record is returned in the journal control area at JCAADATA. The journal record is in CICS transaction storage chained off the TCA of the calling program.

Note: If a direction change occurs, for example, if a GETF follows a GETB, the same journal record will be retrieved.

NOTE
Obtains positioning information for the currently open volume of the specified journal. Positioning data is returned in the journal control area field JCANOTE, and identifies a logical record within a block within a volume. Positioning data includes the volume identification needed for DFHJC TYPE=(OPEN,INPUT) requests that specify SIVOL=YES. The fields used for volume identification of labeled volumes are JCAVOLID, JCAFLG and JCAPRTNO. For unlabeled tape volumes, they are JCARST, JCAVCD, and JCAVSN.

Notes:

1. Positioning data for a journal open for input is returned for DFHJC TYPE=NOTE requests; at least one successful GETB or GETF request must precede the NOTE request.
2. Positioning data for a journal open for output can be obtained by including the NOTE keyword in the output request: for example, DFHJC TYPE=(PUT,NOTE).

POINT
Repositions the currently open input volume to a specified logical record. Before issuing this request, you must load the journal control area field JCANOTE with positioning data returned by a previous NOTE request. Following a successful POINT request, you can retrieve the logical journal record in question with a GETF request.

Note: The correct volume of the journal must be currently open for input, and at least one successful GETB or GETF request issued to it, preceding the POINT request.

EOFADDR=symbolic-address
Indicates the address to which control is to be passed if the journal reaches end-of-file for GETF, GETB or (tape only) POINT requests.

Note: After end-of-file is passed for a tape journal in the forward direction (GETF request), further attempts to retrieve from or reposition the volume will lead to unpredictable results and I/O errors.

IDERROR=symbolic-address
Indicates the address to which control is to be passed if the specified journal does not exist in the journal control table.

INVREQ=symbolic-address
Indicates the address to which control is to be passed if the TYPE of operation is invalid or specifies POINT or NOTE before any reads (GETF or GETB) from the current input volume.

IOERROR=symbolic-address
Indicates the address to which control is to be passed if an I/O error occurs.

JFILEID={SYSTEM|nn|YES}
Identifies the journal referenced in this operation. The default is JFILEID=SYSTEM.

SYSTEM
Specifies the system log.

nn Is a decimal value from 2 through 99, which identifies the journal.

YES
Indicates that the journal identification has been loaded into the JCAJFID field in the journal control area prior to issuing the request.

NORESP=symbolic-address
Indicates the address to which control is to be passed if the requested operation is successful.

NOTOPEN=symbolic-address
Indicates the address to which control is to be passed if the journal is not open.

STATERR=symbolic-address
Indicates the address to which control is to be passed if the journal is open for output, or that the requesting task is not the one with exclusive control.

VOLERR=symbolic-address
Indicates the address to which control is to be passed if a POINT request specifies a volume other than the one currently open for input.

Note: EOFADDR, STATERR, NOTOPEN, VOLERR, IDERROR, INVREQ, IOERROR, and NORESP keywords may be specified in separate DFHJC TYPE=CHECK macros or HANDLE CONDITION requests.

Layout and contents of journal records

Journal data sets are usually created as containing undefined records (RECFM=U). They are then formatted by the journal control program to be compatible with variable-length blocked records (RECFM=VB). When reading a journal, you can defined it as RECFM=VB, and the access method will do the unblocking.

Each block and each record within the block begins with an LLbb length field. Each block contains at least two logical records, because journal control creates a label record as the first record in every block.

SMF formatted journal blocks have a different structure. For information about SMF formatted journal blocks, see "Standard system management facilities block header" on page 400.

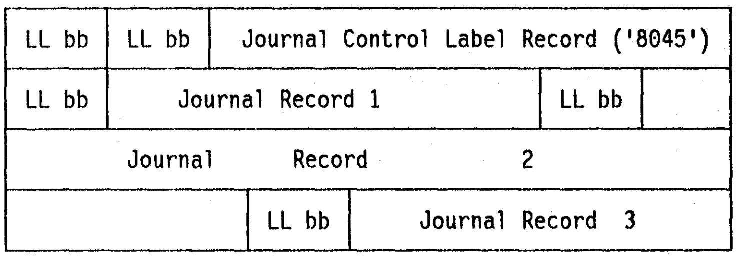


Figure 13. CICS journal format

Each block contains the following logical records:

1. One journal control label record.

   Each block on a journal data set starts with this record. It indicates global CICS information such as the block number, CICS run start time, and journal identification.

2. One or more journal records.

   These records follow the journal control label record, and contain all the information that has been written for the different CICS tasks. These records are variable-length. The number in a particular block will depend on the length of data to be logged, on the size of the journal buffer (as specified in the journal control table), and on the frequency of writes. Writes occur because they have been explicitly forced, or because the buffer is full, or because the "shift-up" value has been passed. The "shift-up" value is a notional record size that is adjusted dynamically to maintain performance.

When retrieved directly from a journal by DFHJC TYPE=GETB or GETF requests, journal records are returned in a CICS transaction storage area pointed to by field JCAADATA and are mapped by the DFHJCRDS DSECT.

The system header (the first 10 bytes) of every journal record, including label records, but excluding DL/I records, consists of these fields:

Field Name in DFHJCRDS DSECT	**Field Size in Bytes**	**Format**	**Contents**
JCRBA	EQU	*	Label for start of journal records
JCRLL	2	Halfword binary	Length of record
JCRBB	2	Binary zeros	Not used
JCRSTRID	2	Hexadecimal	System type-ID
JCRUTRID	2	Hexadecimal	User type-ID
JCRLRN	2	Packed decimal	Record number within block

The system and user type-ID fields, JCRSTRID and JCRUTRID, are the means of distinguishing journal records output by CICS, by such features as automatic journaling, from those issued by direct user requests.

For CICS journal requests, the user type-ID is 0, and the system type-ID consists of a 1-byte function code followed by a 1-byte module code. Valid settings of these codes are contained in the member DFHFMIDS of the CICS assembler-language macro library as shown in Figure 14.

For user journal requests, the system type-ID field always contains binary zeros; the user type-ID field contains the 2-byte hexadecimal code specified by the JTYPEID keyword of the output request.

```
                                                                        DFH00010
                                                                        DFH00020
*********************************************************************** DFH00030
*********************************************************************** DFH00040
* * *                                                             * * * DFH00050
* * *          FUNCTION AND MODULE IDENTIFIERS                    * * * DFH00060
* * *          (SEE FOLLOWING DSECTS: DFHDWEDS,DFHJCADS,DFHJCR)   * * * DFH00070
* * *                                                             * * * DFH00080
*********************************************************************** DFH00090
* *             F U N C T I O N   I D E N T I F I E R S             * * DFH00100
*********************************************************************** DFH00110
*                                                                     * DFH00120
*        X'01' THRU X'7F' ARE RESERVED FOR DL/I                       * DFH00130
*        X'20' PLUS X'8-' ...USE FOR AUTOMATIC JOURNALING             * DFH00140
*        X'40' PLUS X'8-' ...USE FOR AUTOMATIC LOGGING                * DFH00150
*        X'E0' thru X'FF' are reserved for Sync-Point logging         * DFH00160
*              (MUST BE PRESENT IN 'LOGGABLE' DWE'S)                  * DFH00170
*                                                                     * DFH00180
*********************************************************************** DFH00190
* *                         JOURNAL CONTROL                         * * DFH00200
*********************************************************************** DFH00210
FIDJCLAB EQU   X'80'               ...JOURNAL CONTROL LABEL           @ DFH00220
*                                  RECORD (DFHJCR)                      DFH00230
FIDJCLOK EQU   X'81'               'NESTED' LOCK DWE          @BM09013 DFH00240
*********************************************************************** DFH00250
* *             DYNAMIC BACKOUT FUNCTION IDENTIFIERS:-              * * DFH00260
*********************************************************************** DFH00270
FIDDBOFL EQU   X'80'               OVERFLOW DYNAMIC LOG RECORD @BD96830 DFH00280
FIDDBCHN EQU   X'81'               CHAIN DYNAMIC LOG RECORD    @BD96830 DFH00290
*********************************************************************** DFH00300
* *                          FILE CONTROL                           * * DFH00310
*********************************************************************** DFH00320
FIDALOG  EQU   X'40'               ...AUTOMATICALLY LOGGED            @ DFH00330
FIDAJRN  EQU   X'20'               ...AUTOMATICALLY JOURNALLED        @ DFH00340
*                                       PLUS ONE OF...                  DFH00350
FIDFCRO  EQU   X'80'               ...FILE CONTROL READ-ONLY          @ DFH00360
FIDFCRU  EQU   X'81'               ...FILE CONTROL READ-UPDATE        @ DFH00370
FIDFCWU  EQU   X'82'               ...FILE CONTROL WRITE-UPDATE       @ DFH00380
FIDFCWA  EQU   X'83'               ...FILE CONTROL WRITE-ADD          @ DFH00390
FIDFCDSN EQU   X'8F'               ...NEW DSNAME                   @D1A DFH00400
*                                                                       DFH00410
*        NOTE THAT FID* VALUES (AS ABOVE) ARE OFTEN USED BOTH TO        DFH00420
*        IDENTIFY THE FUNCTION OF THE DWE AND THE FUNCTION OF THE       DFH00430
*        LOG RECORD.  IN THE CASE OF THE FIDFC* EQU'S ABOVE, THEY       DFH00440
*        ARE USED FOR LOG RECORDS ONLY.  THOSE   BELOW APPLY ONLY       DFH00450
*        TO DWE'S                                                       DFH00460
*                                                                       DFH00470
FIDFCVWA EQU   X'80'               THIS DWE ADDRESSES A VSWA.      @LBC DFH00480
FIDFCRVY EQU   X'40'               THIS DWE IS ASSOCIATED WITH A ...   *DFH00490
                                   ..RECOVERABLE CHANGE.           @LBC DFH00500
*********************************************************************** DFH00510
*                TRANSIENT DATA FUNCTION IDENTIFIERS:-                * DFH00520
FIDTDIT  EQU   X'F1'               TD DESTINATION'S INPUT TASK        @ DFH00530
FIDTDOT  EQU   X'F2'               TD DESTINATION'S OUTPUT            @ DFH00540
*                                  TASK                                 DFH00550
FIDTDPT  EQU   X'F3'               TD DESTINATION'S PURGE TASK @BM10372 DFH00560
*FIDTDDB EQU   X'08'               ...DYNAMIC BACKOUT MASK    @BI01000 DFH00570
                                                                        DFH00580
                                                                        DFH00590
```

Figure 14 (Part 1 of 5). Journal function and module identifications

```
                                                                         DFH00600
                                                                         DFH00610
FIDTDPLP EQU   X'81'               TD PHYSICAL 'FIRST PUT' LOG         @ DFH00620
FIDTDPGT EQU   X'82'               TD PHYSICAL 'GET' LOG               @ DFH00630
FIDTDPRL EQU   X'83'               TD PHYSICAL QUEUE ZERO LOG          @ DFH00640
*                                    - REUSE=YES                    @LCA DFH00650
FIDTDPLG EQU   X'84'               TD PHYSICAL 'PURGE' LOG             @ DFH00660
FIDTDPQZ EQU   X'85'               TD PHYSICAL QUEUE ZERO LOG       @LCA DFH00670
*                                    - REUSE=NO                     @LCA DFH00680
********************************************************************** DFH00690
*               TEMPORARY STORAGE FUNCTION IDENTIFIERS                 * DFH00700
FIDTSAL  EQU   X'40'               AUTOMATIC LOGGING MASK       @BD5623D DFH00710
*FIDTSDB EQU   X'08'               ...DYNAMIC BACKOUT MASK      @BI01000 DFH00720
FIDTSUPD EQU   X'80'               ..TEMPORARY STORAGE UPDATE   @BD5623D DFH00730
FIDTSPRI EQU   X'F2'               ..TEMPORARY STORAGE          @BD5623D DFH00740
*                                  PURGE/RELEASE                         DFH00750
FIDTSPUT EQU   X'F4'               ..TEMPORARY STORAGE          @BD5623D DFH00760
*                                  PUT/PUTQ                              DFH00770
FIDTSCLN EQU   X'01'               ..TEMPORARY STORAGE CLEAN    @BA21192 DFH00780
*                                  UP DWE                       @BA21192 DFH00790
********************************************************************** DFH00800
*              SPECIAL FEATURES  FUNCTION IDENTIFIERS                  * DFH00810
FIDPSOPC EQU   X'80'               CONTINOUS LOGICAL SPOOLOPEN          FDFH00820
                                                         @E9700T @L5A DFH00830
FIDPSWRC EQU   X'81'               CONTINOUS LOGICAL SPOOLWRITE         FDFH00840
                                                         @E9700T @L5A DFH00850
FIDPSCLC EQU   X'82'               CONTINOUS LOGICAL SPOOLCLOSE         FDFH00860
                                                         @E9700T @L5A DFH00870
FIDPSOPS EQU   X'83'               STANDARD SPOOLOPEN    @E9700T @L5A DFH00880
********************************************************************** DFH00890
*               INTERVAL CONTROL FUNCTION IDENTIFIERS                  * DFH00900
FIDICPDF EQU   X'50'               INTERVAL CONTROL PUT,DEFER   @BM10372 DFH00910
FIDICRGT EQU   X'80'               RESTART GET.                 @BBDI800 DFH00920
*FIDICDB EQU   X'08'               BACKOUT MASK                 @BI01000 DFH00930
********************************************************************** DFH00940
*                PROGRAM CONTROL FUNCTION IDENTIFIERS                  * DFH00950
FIDPCPPT EQU   X'80'               PC REPLACE PPT DEFINITION        @L9A DFH00960
********************************************************************** DFH00970
*                TASK CONTROL FUNCTION IDENTIFIERS                     * DFH00980
FIDKCPCT EQU   X'80'               KC REPLACE PCT DEFINITION        @L9A DFH00990
FIDKCPFT EQU   X'81'               KC REPLACE PFT DEFINITION        @L9A DFH01000
********************************************************************** DFH01010
*              ACTIVITY KEYPOINT FUNCTION IDENTIFIERS:-                * DFH01020
FIDAKS   EQU   X'80'               ACTIVITY KEYPOINT: START.           @ DFH01030
FIDAKE   EQU   X'81'               ACTIVITY KEYPOINT: END.             @ DFH01040
*        EQU   X'82'               RESERVED - WAS FIDKPTCA               DFH01050
*        EQU   X'83'               RESERVED - WAS FIDKPDCT               DFH01060
*        EQU   X'84'               RESERVED - WAS FIDKPTCR               DFH01070
*        EQU   X'85'               RESERVED - WAS FIDKPTST               DFH01080
FIDAKM   EQU   X'86'               ACTIVITY KEYPOINT: MIDDLE.       @L7A DFH01090
********************************************************************** DFH01100
         SPACE 1                                         @ECB1D @L1C DFH01110
                                                                         DFH01120
                                                                         DFH01130
```

Figure 14 (Part 2 of 5). Journal function and module identifications

```
                                                                         DFH01140
                                                                         DFH01150
*********************************************************************** DFH01160
*                 SYNC POINT FUNCTION IDENTIFIERS:-                   * DFH01170
FIDSPUOW EQU   X'EF'                Start-UOWid log record         @D2A DFH01180
FIDSPLUC EQU   X'F0'                LUC RELATED FUNCTION     @EIA5D @L4A DFH01190
FIDLSOSP EQU   X'F1'                LOGICAL START OF SYNC POINT.       @ DFH01200
FIDLEOTK EQU   X'F2'                LOGICAL END OF TASK.               @ DFH01210
FIDPEOTK EQU   X'F3'                PHYSICAL END OF TASK.              @ DFH01220
FIDBEOTK EQU   X'F4'                BAD END OF TASK LOG RECORD  @BM10372 DFH01230
FIDSPR   EQU   X'F5'                SPR LOG RECORD              @BM10372 DFH01240
FIDBEOSP EQU   X'F6'                BAD END OF SYNC POINT LOG   @BM10372 DFH01250
*                                   RECORD                      @BBECB1D DFH01260
FIDRSQ   EQU   X'F7'                REMOTE SESSION QUALIFIER    @BBDI80D DFH01270
FIDSPRMI EQU   X'F8'                RMI DWE. (ALSO USED AS A TEST MASK..*DFH01280
                                    ..FOR THE FOLLOWING RMI CODES.     *DFH01290
                                    ...                     @ECB1D @L1C DFH01300
FIDSPPTC EQU   X'F9'                RMI PREPARE             @ECB1D @L1C DFH01310
FIDSPPTB EQU   X'FA'                RMI ABOUT TO BACKOUT    @ECB1D @L1C DFH01320
FIDSPRSC EQU   X'FB'                RMI RESYNC COMMITTED    @ECB1D @L1C DFH01330
FIDSPRSB EQU   X'FC'                RMI RESYNC BACKED OUT   @ECB1D @L1C DFH01340
FIDSPFGT EQU   X'FD'                RMI FORGET              @ECB1D @L1C DFH01350
FIDSPLTC EQU   X'FE'                RMI 'LOST TO COLD START' @ECB1D @L1C DFH01360
FIDSPNID EQU   X'FF'                RMI 'NOT IN DOUBT'      @ECB1D @L1C DFH01370
*********************************************************************** DFH01380
         SPACE 1                                            @ECB1D @L9P DFH01390
*********************************************************************** DFH01400
*                 RECOVERY CONTROL FUNCTION IDENTIFIERS:-             * DFH01410
FIDRCFWD EQU   X'F0'                FORWARD RECOVERY RECORD        @L9A DFH01420
FIDRCBWD EQU   X'80'                BACKOUT RECORD                 @L9A DFH01430
*********************************************************************** DFH01440
         SPACE 1                                                        DFH01450
*********************************************************************** DFH01460
*                 BMS FUNCTION IDENTIFIERS:-                  @BD5A01J DFH01470
FIDBMPM  EQU   X'81'                ...BMS - PARTIAL MESSAGE ON @BD5A01J DFH01480
*                                                 TEMPORARY STORAGE     DFH01490
FIDBMODS EQU   X'82'                ...BMS - OPEN DATA SET ON   @BD85D1L DFH01500
*                                   BATCH LU                            DFH01510
*********************************************************************** DFH01520
*               TERMINAL CONTROL FUNCTION IDENTIFIERS                 * DFH01530
*                                                             @BD5021J DFH01540
FIDTCML  EQU   X'F0'                SYNC POINT - LOG SEQUENCE   @BD5021J DFH01550
*                                   NUMBERS                             DFH01560
*                         ...THE ABOVE PLUS ANY OF FOLLOW'G 3.. @BM13500 DFH01570
FIDTCDWL EQU   X'01'                ...DEFERRED WRITE DATA      @BD5021J DFH01580
FIDTCFMH EQU   X'02'                ...+ FUNCTION MANAGEMENT    @BD5021J DFH01590
*                                   HEADER                              DFH01600
FIDTCDIP EQU   X'04'                ...+ DIP REQUEST            @BD9960Y DFH01610
*                                                             @BD5021J DFH01620
                                                                         DFH01630
                                                                         DFH01640
```

Figure 14 (Part 3 of 5). Journal function and module identifications

```
                                                                    DFH01650
                                                                    DFH01660
*FIDTCDB EQU    X'08'                ...DYNAMIC BACKOUT MASK      @BD96830 DFH01670
FIDTCAL  EQU    X'40'                AUTOMATIC LOGGING MASK...    @BD5021J DFH01680
FIDTCAJ  EQU    X'20'                AUTOMATIC JOURNALING MASK.. @BM13500 DFH01690
*                          ...THE ABOVE 2 PLUS 1 OF FOLLOW'G SET @BM10372 DFH01700
FIDTCTL  EQU    X'80'                ...SEQUENCE NUMBER ONLY      @BD5021J DFH01710
*                                    (LOG ONLY)                            DFH01720
FIDTCIM  EQU    X'81'                ...INPUT MESSAGE (LOG AND    @BD5021J DFH01730
*                                    JOURNAL)                              DFH01740
FIDTCOM  EQU    X'82'                ...OUTPUT MESSAGE (JOURNAL   @BD5021J DFH01750
*                                    ONLY)                                 DFH01760
FIDTCWP  EQU    X'83'                ...WRITE WAS PURGED (LOG     @BD5021J DFH01770
*                                    ONLY)                                 DFH01780
FIDTCPRR EQU    X'84'                ...POSITIVE RESPONSE         @BD5021J DFH01790
*                                    RECEIVED (LOG ONLY)                   DFH01800
FIDTCIMF EQU    X'85'                ...INPUT MESSAGE (W/FMH,     @BD85D1V DFH01810
*                                    LOG AND JOURNAL)                      DFH01820
FIDTCOMN EQU    X'86'                ...OUTPUT MESSAGE, (W/O      @BD85D1V DFH01830
*                                    FMH, JOURNAL ONLY)                    DFH01840
FIDTCON  EQU    X'87'                ...OUTPUT MESSAGE, FMH,      @BD85D1V DFH01850
*                                    CCOMPL=NO                             DFH01860
FIDTCONN EQU    X'88'                ...OUTPUT MESSAGE, W/O FMH, @BD85D1V*DFH01870
                                     ...CCOMPL=NO                 @BD85D1V DFH01880
FIDTCUA  EQU    X'89'                ...INITIAL TCT USER AREA     @BM10372 DFH01890
FIDTCEIB EQU    X'8A'                ...INITIAL EXEC COMM AREA    @BM10372 DFH01900
*********************************************************************** DFH01910
*                 TABLE BUILDER SERVICES FUNCTION IDENTIFIERS              DFH01920
FIDBSDOP EQU    X'80'                TBS DWE IS 'OPEN'                @LAA DFH01930
FIDBSDCL EQU    X'81'                TBS DWE IS 'CLOSED'              @LAA DFH01940
*********************************************************************** DFH01950
*               GENERAL PURPOSE SUBTASK IDENTIFIERS                        DFH01960
FIDSKDF  EQU    X'80'                ...SK - DEFAULT                  @L6A DFH01970
*********************************************************************** DFH01980
* *               M O D U L E   I D E N T I F I E R S                * * DFH01990
*********************************************************************** DFH02000
*                                                                 @BD5A01J DFH02010
*               (MAY BE X'01'-->X'FF'.)                                    DFH02020
*                                                                          DFH02030
MODIDKC  EQU    X'03'                ...TASK CONTROL                     @ DFH02040
MODIDPC  EQU    X'04'                ...PROGRAM CONTROL                  @ DFH02050
MODIDSC  EQU    X'05'                ...STORAGE CONTROL                  @ DFH02060
MODIDDC  EQU    X'07'                ...DUMP CONTROL                     @ DFH02070
MODIDIC  EQU    X'08'                ...INTERVAL CONTROL                 @ DFH02080
MODIDTC  EQU    X'10'                ...TERMINAL CONTROL          @BD5021J DFH02090
MODIDFC  EQU    X'11'                ...FILE CONTROL                     @ DFH02100
MODIDTD  EQU    X'12'                ...TRANSIENT DATA                   @ DFH02110
MODIDTS  EQU    X'13'                ...TEMPORARY STORAGE                @ DFH02120
MODIDIRC EQU    X'37'                ...IRC INTERFACE          @E211Q @M1A DFH02130
MODIDDL  EQU    X'39'                ...DL/I INTERFACE                   @ DFH02140
                                                                           DFH02150
                                                                           DFH02160
                                                                           DFH02170
                                                                           DFH02180
MODIDBM  EQU    X'40'                ...BASIC MAPPING                    @ DFH02190
MODIDJC  EQU    X'45'                ...JOURNAL CONTROL                  @ DFH02200
MODIDDB  EQU    X'46'                ...DYNAMIC BACKOUT PROGRAM   @BD96830 DFH02210
MODIDVC  EQU    X'47'                ...VOLUME CNTROL PROGRAM @E211Q @L2C DFH02220
MODIDPS  EQU    X'53'                ...SPECIAL FEATURES                FDFH02230
                                                                  @L5A DFH02240
```

Figure 14 (Part 4 of 5). Journal function and module identifications

```
MODIDKPP EQU   X'54'                 ...KEYPOINT PROGRAM                @ DFH02250
MODIDBI  EQU   X'55'                 ...BUILT-IN FUNCTIONS              @ DFH02260
MODIDAKP EQU   X'58'                 ...ACTIVITY KEYPOINT PROG   @BM13500 DFH02270
MODIDSPP EQU   X'59'                 ...SYNC POINT PROGRAM              @ DFH02280
MODIDEIP EQU   X'5A'                 ...EXEC INTERFACE PROGRAM   @BI01102 DFH02290
MODIDTMP EQU   X'5B'                 ...TABLE MANAGER        @EU71T @L3C DFH02300
MODIDSKP EQU   X'5C'                 ...SUBTASK MANAGER              @L6A DFH02310
MODIDRCP EQU   X'CE'                 ...RECOVERY CONTROL PROGRAM     @L9A DFH02320
MODIDTBS EQU   X'E3'                 ...TABLE BUILDER SERVICES.      @LAA DFH02330
MODIDTOR EQU   X'EF'                 ...TERMINAL OBJECT RESOLUTION   @L8A DFH02340
MODIDUSR EQU   X'FF'                 RESERVED FOR USER SYNC     @BD5021J DFH02350
*                                    POINT SUPPORT                        DFH02360
********************************************************************** DFH02370
                                                                          DFH02380
                                                                          DFH02390
```

Figure 14 (Part 5 of 5). Journal function and module identifications

After the common fields (shown on page 165) journal records may be in one of two formats, as follows:

Journal label records

This format applies only to the first record of every block. These are journal management's label records, as follows:

Field Name in DFHJCRDS DSECT	Field Size in Bytes	Format	Contents
JCLRJFID	1	Binary	Journal ID (1-99)
JCLRBLKN	3	Packed decimal	Block number (1-n) in this data set
JCLRVCD	4	Packed decimal	Volume creation date (00yyddd+)
JCLRVSN	2	Packed decimal	Volume sequence number within run (nnn+) (only 1 or 2 for disk journals)
JCLRLBW	4	Binary (disk)	Relative TTR0 of previous block
JCLRTBAL	2	Binary (disk)	Track-balance from previous block (disk journals)
JCLRTIME	4	Packed decimal	Time block written (hhmmsss+)
JCLRRST	4	Packed decimal	Run start time (hhmmsss+)
JCLRDATE	4	Packed decimal	Date block written (00yyddd+)

Other journal records

All other journal records, which are created in response to external requests (DFHJC macro instructions), are continued with from one to three variable-length segments, in this order following the system header:

- System prefix
- User prefix (if any)
- Journaled data.

System header	System prefix	User prefix	Journaled data

Figure 15. CICS journal record format

System prefix: Every journal record includes a system prefix that is variable in length. The system prefix identifies the origin of the record and contains at least the following data:

Field Name in DFHJCRDS DSECT	Field Size in Bytes	Format	Contents
JCSPBA	0	–	Label for system prefix begin address
JCSPLL	2	Halfword binary	Length of system prefix
JCSPFS	3	Binary	Flags (see note at end of table)
JCSPTASK	3	Packed decimal	Task number as in TCAKCTTA
JCSPTIME	4	Packed decimal	Time of request (hhmmsss+)
JCSPTRAN	4	Characters	Transaction identification
JCSPTERM	4	Characters	Terminal identification (or binary zeros)
JCSPREA		EQU *	Label for end of system prefix common root

Notes on JCSPFS: The first two bytes are reserved for future expansion. The third byte is field JCSPF1, containing flags that have the meanings given below. Each EQUATE field shown can be used with JCSPF1 to test the corresponding flag.

JCSPUP	EQU X'01'	User prefix present in record
JCSPSOTK	EQU X'02'	Physical start-of-task
JCSPLSTK	EQU X'04'	Logical start-of-task
JCSPRRIF	EQU X'08'	DFHRUP record in-flight flag
JCSPMIDT	EQU X'10'	Output message in doubt

System prefix additional data: For some CICS journal requests, additional data is included in the system prefix to further identify the originator of the request. This additional data follows the common fields and is usually variable in length; hence the need for the length-field JCSPLL at the start of the system prefix. For journal records created by the CICS file control program's automatic journaling or automatic logging features for file data accesses, the additional data in the system prefix is:

Field Name in DFHJCRDS DSECT	Field Size in Bytes	Format	Contents
JCSPFCFI	8	Character	File identification
JCSPFCRB	4	Character	File control base RBA (ESDS via path only)
JCSPFCRI	1 to 255	–	Record identification

For records journaled to reflect the current data set allocation (JCRSTRID=X'AF'), the system prefix contains the file identification as above, but no record identification. The data following the system prefix is formatted as follows:

```
JCSPFCDL DS    X       1-byte DSNAME length
JCSPFCDN DS    0C      1- to 44-byte DSNAME
```

Note that, in the case of a file name corresponding to a path, the data set name is that of the base cluster.

For journal records created by the CICS terminal control program's automatic journaling or automatic logging features, the additional data in the system prefix is:

Name	Bytes	Format	Contents
JCSPTCVS	4	2 halfwords	VTAM's sequence numbers (2 bytes inbound followed by 2 bytes outbound)
JCSPTCL		EQU *	Label for end of terminal control's prefix

For journal records created by the syncpoint program during intercommunication syncpoint processing, the additional data in the system prefix is:

Name	Bytes	Format	Contents
ORG JCASPREA			
JCAISSQI	2	1 halfword	Sequence number of last inbound syncpoint request
JCAISSQO	2	1 halfword	Sequence number of last outbound syncpoint request
JCAISFL	1	1 byte	Flag
JCAINDT		EQU X'80'	"In-doubt"
JCASSPR		EQU X'40'	Sync point request sent
JCAISAB		EQU X'20'	Successful abort
JCANDTB		EQU X'10'	No DTB if "in-doubt"
JCAIFAIL		EQU X'08'	Session failed
JCAISOP		DS CL3	Operator identification
JCAISTM		DS CL4	Intersystem terminal identification
JCAISSPL		EQU *-JCASPBA	Intersystem communication system prefix length

User-prefix: The user prefix is optional, and is placed in a journal output record next to the system prefix, in response to the PFXADDR and PFXLGTH keywords of the journal control output request. As with the system prefix, the user prefix always begins with a halfword binary length field; the data indicated by the PFXADDR keyword follows the halfword. For journal records that include a user prefix, the flag byte JCSPF1 of the system prefix has the indicator bit JCSPUP set to one.

Journaled data: The final segment of a journal record is the data, as specified by keywords JCDADDR and JCDLGTH of the journal control output request. The length of the data portion of a journal record can be computed by subtracting from the length of the journal record (JCRLL) the length of the record prefix (10 bytes) and the length of the system prefix (JCSPLL) and the length of the user prefix (in the field, if any, defined by yourself).

Journal records for DL/I

For records written to the CICS journal on behalf of DL/I, the records will contain the following data:

Field Name in DFHJCRDS DSECT	Field Size in Bytes	Format	Contents
JCRLL	2	Halfword binary	Length of record
JCRBB	2	Binary zeros	not used
JCRDLIRC	V	Data	Variable length of DL/I record

Reading journal data sets

CICS can read journals online and in its offline utilities.

This section describes each of these methods.

Note: When disk journal data sets are opened at system initialization, the pointers are positioned so that the output will continue immediately after the last record written to the journal. When reading the journals, you must be aware that data may be present for more than one CICS execution. If this is not required, the journal data sets must be formatted before being used for output. As an alternative, with the data set offline, you can force an EOF into the block number 1 position. However, for system log and DL/I, or if you are using DFHJ01X, this may not be true. For more information about these combinations, see the *CICS/MVS Recovery and Restart Guide*.

Reading journal data sets during CICS execution

Journals are designed to be heavily used, shared output files, and are normally opened for output at system initialization.

Provision is made for reading journals online; the data can be read either forward or backward. To read a journal, a task must first close the journal, at which time that task is given exclusive control of the journal, which means that access by other tasks will not be permitted. Exclusive control is released when the same task reopens the journal for output. If the task that owns the journal requests a write, control will be returned with an invalid-request condition. If any other task requests a write to the journal, that task will be put in a wait state until the journal is available for output. Other tasks will not be allowed to read the journal.

It is your responsibility to release exclusive control of a journal by opening it for output. To ensure that this is done in case of abnormal termination of the controlling task, you should establish an abend exit routine for the task using the DFHPC TYPE=SETXIT macro instruction or a HANDLE ABEND command. The exit routine should restore the journal to output status.

Before initiating a task that is expected to retain exclusive control of a journal for more than a few seconds, you should plan to disable any other transactions that might issue requests to that journal. Transactions can be disabled and enabled using the EXEC CICS SET TRANSACTION command (see page 459) or CICS master terminal facilities (see the *CICS/MVS CICS-Supplied Transactions* manual).

Because the format of journal tapes is compatible with that of extrapartition data sets, it is possible to read journal volumes using the transient data facility, provided that the necessary entries have been added to the destination control table.

Reading a journal backward

Certain functions may require access to a few journal records that were written in the preceding minutes of operation. The purpose of this action is usually corrective, such as backing out updates to the database by a task that subsequently terminated abnormally. The records to be retrieved would probably be the before-image of database records that were written to the system journal by the automatic journal feature. Since this type of operation is likely to retain exclusive control of a journal for only a few seconds, it is unlikely that you would want to disable other transactions that issue requests to that journal. The sequence of events for this application might be as follows:

1. A DFHJC TYPE=GETJCA macro instruction is issued to acquire a journal control area for the input records.
2. A DFHJC TYPE=CLOSE,JFILEID=SYSTEM macro instruction is issued to close the journal and give exclusive control to the requesting task. If the journal is on tape, LEAVE=YES is also specified so that the file will remain correctly positioned after the last output block.
3. A DFHJC TYPE=(OPEN,INPUT),VOLUME=CURRENT,JFILEID=SYSTEM macro instruction is used to open the journal for input, using the current tape volume or disk data set. This also implies that the journal is to be read backward beginning with the last output block.
4. DFHJC TYPE=GETB,JFILEID=SYSTEM,EOFADDR=address macro instructions are issued to read the journal records in reverse chronological order. Note that an attempt by this task to update the database at this time could initiate a request for automatic journaling that, in turn, would return an invalid request condition because the system journal is closed for output. Instead, journal records to be used for later updating can be retained on the transaction's storage chain. Other journal records are discarded by issuing a DFHSC TYPE=FREEMAIN macro instruction.

   When the beginning of a tape reel or a disk data set is encountered while reading backward, an end-of-file condition is indicated. Your end-of-file routine should switch to the preceding volume or data set by issuing the following macro instructions:

   ```
   DFHJC    TYPE=CLOSE,JFILEID=SYSTEM
   DFHJC    TYPE=(OPEN,INPUT),VOLUME=PREVIOUS,JFILEID=SYSTEM
   ```

   The positioning is again after the last output block on the volume or data set. If there is no previous volume, a VOLERR condition code is returned.

Note: For disk journals, the one or two data sets specified are periodically reused. An attempt to read backward so far that logical wraparound occurs will usually result in an I/O error. The unlikely case that an I/O error does not occur can be detected by a sequence break in the time-and-date stamp in the journal record prefix.

5. A DFHJC TYPE=CLOSE,JFILEID=SYSTEM macro instruction is issued to close the system journal for input after all desired records have been read.
6. A DFHJC TYPE=(OPEN,OUTPUT), VOLUME=CURRENT,JFILEID=SYSTEM macro instruction is issued to release exclusive control of the system journal and make it available for output. If the journal is on disk, the data set is positioned after the last record written. If it is on tape, the VOLUME=CURRENT is ignored and output resumes with a new reel.

   The task can now process the records retained in step 4.

Reading a journal forward

Some application programs need to read large numbers of journal records. These application programs would typically take more than a few seconds to execute, and would therefore only be practical if the journal is on tape and is not being accessed by any other task. The tape volumes being read would probably have been written and closed previously.

The appropriate means for fulfilling such a need differ somewhat, depending on how the journal was defined. In any case, the system programmer should make sure that, when the journal is being written, journal control NOTEs are taken at suitable times and kept for use in reading the journal. The format of a NOTE depends on the labeling of the journal, but the offsets of significant fields are always the same.

Where unlabeled tapes are used, it may be convenient to define a separate journal entry for use by the application program that reads them. Where the journal specifies standard labeled tapes, CICS will control the selection of tape volumes to be mounted, in accordance with its stored volume descriptors. Thus, the introduction of such an "alias" journal control table entry will not work, and it will not be possible to read and write simultaneously on one journal, even though the volumes are known to be distinct.

For example, assume that an application program is to read previously written reels of a journal that is defined as JFILEID=13. The sequence of events considered here for this application program might be as follows:

1. A DFHJC TYPE=GETJCA macro instruction is issued to acquire a journal control area for the input records.
2. A DFHJC TYPE=CLOSE,LEAVE=NO,JFILEID=13 macro instruction is issued to close the journal; the task is also given exclusive control of the journal. LEAVE=NO causes the current output reel to be rewound and unloaded. Note that this journal, as all other journals, is opened for output at system initialization, unless OPEN=DEFERRED has been specified in the journal control table.

3. The application fills in the NOTE field in the JCA. A DFHJC TYPE=(OPEN,INPUT),VOLUME=NEXT,SIVOL=YES,JFILEID=13 macro instruction is issued. VOLUME=NEXT causes the volume to be positioned to read forward, beginning with the first block. SIVOL=YES uses the NOTE fields to decide which volume to open.

   DFHJC TYPE=POINT can be used to position the current volume to the record previously NOTEd.

4. DFHJC TYPE=GET,EOFADDR=addr,JFILEID=13 macro instructions are issued to read the journal forward. Each request retrieves the next logical record. If an end-of-file is encountered and more records are to be read by the task, the following macro instructions are issued in the end-of-file routine:

   ```
   DFHJC    TYPE=CLOSE,JFILEID=13
   DFHJC    TYPE=(OPEN,INPUT),VOLUME=NEXT,JFILEID=13
   ```

5. When all desired records have been read, a DFHJC TYPE=CLOSE,JFILEID=13 macro instruction is issued to close the journal for input.

6. A DFHJC TYPE=(OPEN,OUTPUT),VOLUME=CURRENT,JFILEID=13 macro instruction is issued to release exclusive control of the journal and make it available for processing by other tasks, or for the system to close it at system termination. This action will open a new tape volume and will write a label on it.

All journals entered in the journal control table are normally opened for output during system initialization. You can defer the opening of selected journals by specifying OPEN=DEFERRED in the journal control table. This can be used to allow a user program to open a journal for input to read the records written during a previous execution of CICS. You may want to execute this program during postinitialization processing by entering it in the appropriate program list table (PLT). When the deferred open option is used, the program that first opens the journal must issue a special form of the DFHJC macro in place of the normal DFHJC TYPE=GETJCA. It is:

```
DFHJC    TYPE=(GETJCA,OPEN),VOLUME=FIRST,
         JFILEID=nn,NORESP=symbol
```

This macro gives the requesting task exclusive control of the journal, acquires a journal control area, and collects the current data set pointer information if a disk file is referenced. You can then issue a DFHJC TYPE=OPEN macro instruction for input or output, current or previous volume according to the conventions described in "Opening a journal volume or data set — DFHJC TYPE=OPEN" on page 158.

Reading journal data sets offline

The information in this section is presented in terms of:

- How to write an offline program to read the journal data sets
- Using the offline program.

Writing the offline program

Journal data sets can be read by user-written offline programs. Although written as operating-system undefined (U-format) records by CICS journal management, the blocks are compatible with records of the variable length blocked (VB) format. Each block begins with a 4-byte block-length field ('LLbb'), and each logical record within a block begins with a 4-byte record-length field ('LLbb'). The data set label information will indicate U-format, but this can be overridden to VB (by a DD statement), so that data management will deblock records, and will provide them to the offline program.

When using standard labeled tapes, a user standard header and trailer record will be written at the start and end respectively of each volume. (These will be bypassed, depending upon the JCL options, when the journal is opened or closed.) However, you can read these labels by using the standard exit routines on open or close.

The layout of these 80-byte records is shown below and is available in the DSECT DFHTULDS.

Name	Size	Format	Contents
TULHDR	4	Character	Identifier of label (UHL1 or UTL1)
TULSERS	8	Character	Name of series
TULPART	8	Zoned decimal	Part number within series
TULDATE	5	Zoned decimal	Date (YYDDD)
TULTIME	7	Zoned decimal	Time (HHMMSSS) when label was written
TULCND	1	Character	Condition of volume
TULSUCKN	1	Character	Validity of successor
TULCHAIN	6	Character	Blank or next/previous volume identifier
Reserved	34		
TULIDENT	6	Character	Identifier of this volume

In the header label, TULCHAIN always correctly names the previous volume, except that when TULPART is "1", there is no previous volume.

If, in the trailer label, TULSUCKN is "T", the successor volume has been selected, but not yet opened. So a failure after the label was written could cause its TULCHAIN value to be wrong. If TULSUCKN is "D", the next volume has been used. That is, the next volume is that named in TULCHAIN.

You should be aware that unless a journal volume was successfully closed when last written during CICS execution, or had a tapemark written by the DFHTEOF program, there will be no end-of-file indicator on the volume, and data may run into old records and wrongly formatted blocks. This is shown in the following table:

Close of Journal	State
Normal	UTL1 present
Abnormal	No UTL1, possibly bad records
Abnormal and repaired by TEOF	No UTL1

Offline user-written programs can map journal records by issuing the DFHJCR CICSYST=YES statement, which results in the DFHJCRDS DSECT being included in the program from the CICS assembler-language macro library. The DSECT thus generated is identical to that obtained for CICS programs by the COPY DFHJCRDS statement, except that the fields are not preceded by a CICS storage accounting area. The DSECT is intended to map journal records directly in the block, rather than in a CICS storage area (see "Reading journal data sets during CICS execution" on page 174).

Using the offline program

The offline program can be executed against a DISK or TAPE journal device. The following points should be considered when reading journals while CICS is still active.

- For a DISK journal, two data sets should be allocated. The appropriate JCT option (JOUROPT=PAUSE) and JCL statement (DISP=SHR) must be specified. The JCL for the offline batch program must also be written. You are responsible for ensuring that journal volumes are read in the required sequence, by concatenating DD statements in the correct order.
- For a TAPE journal, the journal volume can be removed and read whenever you want. Another tape volume can be mounted to record data while the first volume is being processed. The advantages of a tape journal over a journal on a disk device are that the job to read the tape journal can run for a relatively long time and is usually easier to process clerically because there is no need to alternate between the information on the two separate data sets.

Printing journal files

You can use the CICS provided utility, DFHJUP, to print journal files.

User-replaceable modules

The user replaceable DFHXJCO and DFHXJCC modules are called when a **disk** journal is opened for output and when it is subsequently closed after output. DFHXJCO is called on opening, and DFHXJCC on closing. You can write your own modules to replace the CICS-provided ones. You can then carry out whatever processing you want to do at journal open or close time. DFHXJCO and DFJXJCC can be invoked during initialization before CICS resources are available and before high-level language initialization is complete. For these reasons, they must be Assembler macro-level programs and should only perform simple functions.

Some possible uses are:

- Control procedures to ensure that journal data sets have been archived before they are made available for reuse.
- Automatic submission of journal archiving jobs through an internal reader, rather than using manual procedures. This is particularly valuable in an XRF environment, where the need to minimize operator involvement in the recovery procedure may be an important factor.
- Dynamic allocation of journal data sets to avoid the need to archive them immediately before they may be reused.

DFHXJCO is invoked for OPEN,OUTPUT calls. It receives control just before the open request is passed to the journal subtask.

DFHXJCC is only invoked if the journal is in output mode. It receives control just after the close request has been completed by the subtask, whether the request was successful or not. If DBRC is operating, the module receives control before the call to DBRC that indicates the closing of the log data set.

The versions of DFHXJCO and DFHXJCC that are supplied merely return to their callers. If you modify the modules so that they can carry out your own processing, note the following points:

- Do not use interval control requests within DFHXJCO or DFHXJCC, because this may cause any tasks that require journaling services to wait.
- You must not alter registers 12 and 13, which point to the TCA and CSA on entry.
- You cannot influence the journal flow with return codes from these modules.
- Apart from registers 12 and 13, no parameters are passed to these modules.
- Because these modules may be invoked during initialization, the same considerations regarding CICS resources apply as for system initialization overlays. For the same reason, you must code in Assembler, because high-level languages may not have been initialized before these modules are invoked.

  You may need to consider using the fields CSAXST and CSASSI2 to verify initialization processing.
- The module can discover the journal which is being processed by looking in the JCA. Not all fields in the JCA are guaranteed from release to release, so the module should only check those fields that are needed for the identification of the journal. This includes field JCAJCTTE which provides the JCTTE address. Using this address, confirm that JCAJFID matches the journal number in the JCTTE.
- During emergency restart, when data set DFHJ01X is to be used, the most recently used data set of DFHJ01A/DFHJ01B is opened and closed to ensure that it has an end of file marker. This is achieved by using OPEN VOL=FIRST, followed by a close, before the OPEN VOL=EMEREXT that opens DFHJ01X for output. These two opens and the close call DFHXJCC and DFHXJCO in the normal way. You should be aware that the OPEN is

done only to allow the CLOSE. Your code should take account of this special case.

- During restart, if the journal is within 3 tracks of end-of-volume, CICS does not open the journal, does not call DFHXJCC, but ***does*** call DFHXJCO. In this case, archiving functions in DFHXJCO are carried out, but functions in DFHXJCC are omitted.

Part 4. Devices and telecommunication access methods

This part describes your role in providing support for terminals and access methods, such as the use of VTAM with logical units, and the TCAM interface (both SNA and non-SNA) to CICS/MVS 2.1.2. It also describes how to initial program load (IPL) the System/7.

"Chapter 4.1. ACF/VTAM logical units with CICS" on page 185 describes the functions you can use to implement and maintain CICS features for logical units.

"Chapter 4.2. The CICS/TCAM interface" on page 205 gives information on implementing the TCAM interface to CICS, allowing CICS to run as an application program under TCAM.

"Chapter 4.3. Writing a transaction to IPL the IBM System/7" on page 229 provides information on how to write a transaction to IPL the System/7 on start/stop and BSC lines.

"Chapter 4.4. IBM 3735 Programmable Buffered Terminal" on page 231 contains information on the system generation and table preparation specifications required to generate support for the 3735 Programmable Buffered Terminal.

"Chapter 4.5. IBM 3740 Data Entry System" on page 233 gives similar information on the 3740 data entry system.

"Chapter 4.6. IBM 3600 Finance Communication System in a BSC network" on page 235 gives similar information on 3600 BSC devices.

"Chapter 4.7. Modifying the terminal control table" on page 237 provides reference information on the macro and operands of the terminal control macro instruction interface (DFHTC CTYPE macros).

"Chapter 4.8. The user program for automatic installation of terminals" on page 249 describes how to control and monitor terminals logging on to CICS.

"Chapter 4.9. Exits for "terminal-not-known" condition" on page 277 describes two exits that enable a solution to the terminal-not-known condition, which can occur when intercommunicating regions use both SHIPPABLE terminals and automatic transaction initiation (ATI).

Chapter 4.1. ACF/VTAM logical units with CICS

In a systems network architecture (SNA) teleprocessing network, the remote work station is not always simply a terminal. More often, it is one of several terminals attached to a terminal controller. Furthermore, the terminal controller may contain one or more user-written programs. In SNA terminology, however, the remote entity with which the CICS application program communicates is always a **logical unit**. This chapter provides a general description of functions available for implementing and maintaining CICS features for logical units.

See the appropriate CICS subsystem guide for a full discussion of the logical unit being used. These guides are:

- *CICS/OS/VS IBM 3270 Data Stream Device Guide*
- *CICS/OS/VS IBM 4700/3600/3630 Guide*
- *CICS/OS/VS IBM 3650/3680 Guide*
- *CICS/OS/VS IBM 3767/3770/6670 Guide*
- *CICS/OS/VS IBM 3790/3730/8100 Guide*.

Users of the extended recovery facility should read the *CICS/MVS XRF Guide* for a description of the special considerations that apply to ACF/VTAM logical unit support in an XRF environment.

Overview of system programmer requirements

The system programmer responsible for logical units in a CICS environment has four main divisions of responsibility:

- Generating an advanced communications function network control program/virtual storage (ACF/NCP/VS) to control the transfer of data between the host processor and the nodes of the logical unit teleprocessing network. The ACF/NCP/VS resides in a communications controller. Because CICS does not interface directly with the ACF/NCP/VS, this chapter contains no information concerning ACF/NCP/VS generation. See the *NCP and SSP Generation and Load Guide*, SC33-3348.
- Defining an ACF/VTAM system that supports telecommunications within the CICS subsystems. A brief discussion of the ACF/VTAM definition procedure related to system programming functions is presented in this chapter. A general description of the ACF/VTAM definition procedure is given in *Network Program Products Planning*, SC23-0110.
- Defining a CICS system that supports the subsystem hardware configuration and desired programming configuration. This chapter discusses this requirement, but it describes only the modifications and additions to CICS system programming functions that relate to CICS subsystem support.
- Correctly configuring the Synchronous Data Link Control (SDLC) terminal controller and writing the necessary programs to control the terminals that are attached to it, and that are to communicate with CICS.

When planning for CICS support of logical units under ACF/VTAM, you have to bear in mind the requirements of:

- ACF/VTAM support for the logical units.
- Connection, input, and output services.
- Basic mapping support (BMS) services for the appropriate devices.
- The node abnormal condition program (DFHZNAC), the function of which is to handle abnormal situations involving a logical unit and to allow you to generate the node error program (NEP) to perform error handling.
- Message option groups (to be referenced by the program control table (PCT) entry for a task) that permit certain processing and logging characteristics to be associated with particular transactions.
- A terminal control macro interface that provides additional system programming capabilities.
- The option to code user exit-routines to be activated during processing of a request by the terminal control management module (DFHZCP).
- Collecting statistics that can be used for system tuning.
- Message switching facilities for certain logical units.

The explanations of these facilities and of related concepts involving CICS system programming responsibilities in an ACF/VTAM network are discussed below.

"Chapter 4.2. The CICS/TCAM interface" on page 205 provides information on system programming responsibilities in a TCAM SNA network. You can also consult "Chapter 4.8. The user program for automatic installation of terminals" on page 249. For details of the commands used for automatic installation of terminals, see the *CICS/MVS Resource Definition (Online)* manual. For macro information, see the *CICS/MVS Resource Definition (Macro)* manual.

Basic concepts

You should understand several concepts and facilities that are basic to your involvement in generating and maintaining CICS support of logical units. They are:

- An additional terminal control program module (DFHZCP) to support ACF/VTAM services
- ACF/VTAM indicators (SNA commands) and responses
- Communication with logical units.

Terminal control program dual module generation

ACF/VTAM is the required access method interface between CICS and logical units. The non-ACF/VTAM terminal control program (DFHTCP) does not provide the required support for ACF/VTAM capabilities; ACF/VTAM support is available only through the CICS ACF/VTAM terminal control programs.

DFHTCP and the ZCP group of programs are two separate collections of modules, generated when DFHSG PROGRAM=TCP is specified. They are always assembled separately and loaded separately. The ZCP group of programs should always be generated, even for a non-ACF/VTAM system, because it contains some internal routines that are necessary for the successful operation of DFHTCP. ACF/VTAM support within the ZCP group is generated by specifying ACCMETH=VTAM in the DFHSG PROGRAM=TCP macro instruction; the VTAMDEV operand of this macro instruction controls any device-dependent code that must be generated within the ZCP group for the ACF/VTAM-supported logical units under CICS. The ACCMETH and VTAMDEV operands must be specified to provide support for CICS logical units under ACF/VTAM.

The TCP and ZCP operands of the DFHSIT TYPE=CSECT macro instruction specify the suffixes of DFHTCP and the ZCP group, respectively, to be loaded by the system initialization program (DFHSIP). If TCP=NO is specified, no DFHTCP load is performed. In contrast, there is no ZCP=NO option to suppress the ZCP group. This is because, as explained earlier, the ZCP group is always generated with DFHTCP, whether or not ACF/VTAM support is subsequently generated.

ACF/VTAM indicators

The ACF/VTAM indicators that are available for your use are described under DFHTC CTYPE=COMMAND in "Chapter 4.7. Modifying the terminal control table" on page 237.

A CICS interface with terminal control allows you to write routines that request the sending of SNA data flow control commands from CICS to the application program of certain logical units. For example, a function provided by an ACF/VTAM indicator may be needed in the installation's error recovery routine (DFHZNEP). In the case of certain logical units, you should use the indicator interface (DFHTC CTYPE=COMMAND macro) to request an ACF/VTAM function, rather than directly altering bits in the TCTTE. Any direct changing of bits leads to unpredictable results if any future changes are made in the TCTTE internal structure.

ACF/VTAM indicators are always sent by CICS with definite function-management-end (FME/DR1) response requested, whether they are sent on behalf of your request or a CICS management module request. CICS DFHZCP calls the appropriate routine and returns control to the requester when the response is received.

ACF/VTAM indicators are also used by CICS management modules. You should thoroughly understand each indicator before using it. You should also understand how and when they are used by CICS, because the misuse of any of them can lead to unpredictable results.

Connection services

Before any communication between CICS and the logical unit can occur, CICS must first be connected to ACF/VTAM. The CICS system initialization program (DFHSIP), which comes in a pregenerated version, or which you can generate using the DFHSG PROGRAM=CSO macro instruction, issues the appropriate ACF/VTAM macro instruction to open the CICS access method control block (ACB) to make the connection. This identifies CICS to ACF/VTAM as one of its application programs. Only then can ACF/VTAM carry out a request to connect a logical unit to CICS and thus allow communication between these two nodes. A logical unit that is connected to CICS is said to be owned by CICS for the duration of the connection.

The CICS APPLID operand provides the name that DFHSIP uses when opening (and closing) its ACB to define itself to ACF/VTAM during CICS system initialization, or by using the master terminal dynamic open facility for the ACB. You specify the APPLID operand in the DFHSIT macro instruction. Only one APPLID chosen in this manner is used by CICS per initialization. Any name coded with the APPLID operand must have been defined during ACF/VTAM definition using ACF/VTAM's APPL statement.

In addition, the dynamic close of the ACF/VTAM ACB facility allows CICS to continue, even though ACF/VTAM may not be operational at the time.

To build the access-method-dependent portions of the TCT for ACF/VTAM support, you must specify ACCMETH=VTAM with the DFHTCT TYPE=INITIAL macro instruction. Resource definition online (CEDA) is only used for VTAM terminals, so there is no need for this option in CEDA. If ACCMETH is omitted from your macro, the default value is NONVTAM. The former default value of BTAM is, however, still valid. Specifying either NONVTAM or BTAM permits existing TCTs to be assembled without change. You may turn off run-time use of VTAM by coding VTAM=NO in the SIT.

You control the connection services available for a particular logical unit using the CEDA DEFINE TERMINAL or the CEDA DEFINE TYPETERM commands or using the DFHTCT TYPE=TERMINAL macro instruction. ACCMETH=VTAM must be specified to create the necessary ACF/VTAM TCTTE for each logical unit. For each TCTTE, CICS automatically creates an accompanying node initialization block (NIB) by issuing the ACF/VTAM NIB macro instruction. The NIB is used to convey several operating parameters that apply to the connection being established.

The NIBs are used to process OPNDST requests, and are not involved during normal logical unit I/O processing. The ACF/VTAM OPNDST request causes ACF/VTAM to establish the connection between CICS and the logical unit. This connection is called an SNA session, and is completed when the logical unit sends a positive response to the SNA BIND command sent by ACF/VTAM as the result of the OPNDST request.

This works in the following way:

1. The BIND sets out the protocol used between CICS and the logical unit.
2. The BIND offers this protocol, in the form of a set of parameters, to CICS.
3. CICS can accept or alter these parameters.
4. Then the logical unit can accept or reject them in the form in which CICS left them.
5. If the parameters are accepted by the logical unit, the connection is complete.

In some cases, the logical unit also has the chance to alter the parameters. Then CICS has to approve the parameters in a second pass. In this case, it is more likely that the connection will not be made because both sides have to approve the other's alterations.

The NETNAME operand of the CEDA DEFINE TERMINAL command or the DFHTCT TYPE=TERMINAL macro instruction provides the symbolic name for the logical unit by which it is known throughout the network. This is the name that CICS specifies to identify the logical unit that is represented by this TCTTE in CICS. The same symbolic name must also be defined to ACF/VTAM during ACF/VTAM system definition using the logical unit macro instruction and to the ACF/NCP/VS during ACF/NCP/VS generation. The NETNAME must be unique for each terminal defined to CICS.

Logon

After CICS has been connected to ACF/VTAM, any logon requests for CICS are passed to CICS (unless the MACRF=LOGON operand of the ACB macro instruction was specified during ACF/VTAM definition, in which case ACF/VTAM is not allowed to queue any logon requests for CICS). In general, the CICS logon exit is scheduled by ACF/VTAM in response to a request initiated either by CICS, or by a logical unit.

To specify that a simulated logon is to be performed for a particular logical unit, you must specify the AUTOCONNECT option of the CEDA DEFINE TYPETERM command or the CONNECT operand of the DFHTCT TYPE=TERMINAL macro instruction. When CICS issues the ACF/VTAM SIMLOGON macro instruction in response to an AUTOCONNECT or CONNECT specification, it also supplies the address of the particular request parameter list (RPL) that contains the address of the NIB whose NAME field identifies the logical unit for which the simulated logon request is to be performed. This drives the logon exit logic in DFHZCP to establish connection with the logical unit. If you do not specify AUTOCONNECT or CONNECT, or if you code the INSERVICE(NO) option of the CEDA DEFINE TERMINAL command or TRMSTAT='OUT OF SERVICE' then the logical unit is not connected to CICS at system initialization, but awaits either a master-terminal operator connection request, a logical unit logon initiated by the ACF/VTAM network operator, or a terminal operator logon from a logical unit.

CICS logical unit support provides a RELREQ exit-routine so that any other ACF/VTAM applications wishing to use a logical unit currently owned by CICS can indicate their needs. When no more work is available for the requested logical unit, CICS checks whether it is permitted to release it. (The RELREQ and DISCREQ options of the CEDA DEFINE TYPETERM command or the RELREQ operand of the DFHTCT TYPE=TERMINAL macro instruction defines whether or not a logical unit can be released by CICS.) If it can be released, the existing connection is broken.

Conversely, CICS can also request the use of a logical unit currently owned by another ACF/VTAM application program. For example, a SIMLOGON is always performed with the RELREQ and RPL options so that CICS can indicate its need of the logical unit to any ACF/VTAM application program that currently owns it.

Input services

Input services handle both data from the logical units and asynchronous input such as ACF/VTAM indicators. This section describes CICS data input in general. More specific information on ACF/VTAM indicators was provided under the heading "Basic concepts" on page 186.

CICS receives user data into the system at two different times. The first is when data is entered to create a new user transaction. The other is in response to a CICS application program request for data from a logical unit. To satisfy these two different situations, CICS uses two distinct kinds of ACF/VTAM RECEIVE macro instructions.

To obtain transaction-originating data, CICS puts all logical units that have no tasks attached into the ACF/VTAM continue-any state. The ACF/VTAM RECEIVE macro instructions with the OPTCD=ANY operand are then issued by CICS to allow any data entered by the logical unit to be received. You control the number of receive-any macro instructions issued by specifying the number of RPLs to be generated. The RAPOOL operand of the DFHSIT macro instruction is used to specify the fixed number of RPLs that are generated in the TCT prefix.

CICS issues a receive-any macro instruction for each RPL not currently in use, except when the short-on-storage condition is present. In this case, the receive-any buffers are temporarily released and no receive-any requests are issued.

The number of RPLs required is dependent on the expected activity of the system, the average transaction lifetime, and the maximum-task value specified. To aid you in choosing a size for the RPL pool, CICS keeps a count of the maximum number of RPLs that were satisfied at any one dispatch scan, plus how many times this maximum was reached. (See "Statistics" on page 202.)

Associated with each receive-any RPL is an I/O area, the size of which is
| specified by the RAMAX operand of the DFHSIT macro instruction. If you are
| using LU6.2 communications, the minimum RAMAX value is 256. If the input length exceeds the size of this I/O area, ACF/VTAM gives CICS only as much of the data as fits into the CICS I/O area, and tells CICS how much was received in total. CICS then handles the data in one of two ways:

- If chain assembly was not requested, CICS obtains, through its storage control program, an area large enough to accommodate the data, and then receives the rest of the input data that was kept by ACF/VTAM, because all of it did not fit into the CICS I/O area.

- If chain assembly was requested, and the total data length exceeds the second operand of the IOAREALEN|TIOAL parameter, an exception response is sent by the node abnormal condition program (DFHZNAC) to the terminal.

The first value in the IOAREALEN option of the CEDA DEFINE TYPETERM command, or in the TIOAL operand of the DFHTCT TYPE=TERMINAL macro instruction, is provided so that you can specify the minimum size of a TIOA for a particular TCTTE. If you are using LU6.2 communications, the minimum TIOA size is 256. If IOAREALEN or TIOAL is not specified, there are no minimum size requirements, and the input data is sent in a TIOA, suitably rounded up to equal the length of the data, to the CICS application program. Therefore, if IOAREALEN/TIOAL is greater than the length of the input data, the length specified in IOAREALEN/TIOAL is passed to the application program regardless of the actual length of the input data.

To obtain data in response to a read requested by a CICS application program, DFHZCP issues an ACF/VTAM RECEIVE macro instruction with the OPTCD=SPEC operand to allow data from a specific logical unit to be received. The data is received directly into the user TIOA; no separate receive-specific I/O area is provided.

Each input message is called a chain. If its length exceeds the maximum output buffer size for the terminal (RECEIVESIZE option of the CEDA DEFINE TYPETERM command or the RUSIZE operand in DFHTCT TYPE=TERMINAL), the message will be broken up into a series of links (or request units) that do not exceed this buffer size. The ACF/VTAM RECEIVE macro obtains only one request unit at a time. You can control whether a CICS application program input request is to be satisfied by a single request unit or by the assembled chain of request units. This control is provided by the BUILDCHAIN option of the CEDA DEFINE TYPETERM command or the CHNASSY operand of the DFHTCT TYPE=TERMINAL macro.

Output services

Output services handle both data sent to the logical units and asynchronous output such as ACF/VTAM indicators and commands. This section describes CICS support of data output in general. More specific information on ACF/VTAM indicators was provided under the heading "Basic concepts" on page 186.

When generating support for the available output services, you have the following main areas of responsibility:

- Determining the maximum data length that each logical unit can receive. This value is specified by the SENDSIZE option of the CEDA DEFINE TYPETERM command or the BUFFER operand of the DFHTCT TYPE=TERMINAL macro instruction, except for 3270 logical units, which use segmenting with the BUFFER size set to zero. If a message longer than this value is to be sent, it is broken into as many links (request units) as necessary. Each link has a maximum size equal to the BUFFER value. The first link may contain the SNA function management header (FMH), but the total length of this first link (including the FMH) does not exceed the value of BUFFER. The default value is zero, which specifies that the data should not be chained, but should be sent just as it is presented to DFHZCP by the CICS application program.

The value specified in the BUFFER operand must not exceed the logical-unit buffer size minus the buffer prefix size, as specified to ACF/NCP/VS. For information on specifying the buffer prefix size, see the *IBM 3704 and 3705 Communications Controller Network Control Program Generation and Utilities Guide and Reference Manual (for OS/VS and VSE ACF/VTAM) Users*, GC30-3008.

Specifying the response level to be used by CICS when transmitting user data. You can specify either:

- Function management end (FME), also known as definite response type 1 (DR1), or:
- Reached recovery node (RRN), also known as definite response type 2 (DR2).

This is specified in the RESP operand of the DFHTCT TYPE=INITIAL macro instruction, and applies to 3600 logical units. This level is used for both normal and exception response requests. The default value is FME. All other logical units use DR1 only, except LU6.1 and LU6.2 which use DR2 for synchronization purposes.

Message recovery and emergency restart

There are two types of failure that require message recovery:

- Catastrophic failures
- Noncatastrophic failures.

A catastrophic failure is one in which either CICS abnormally terminates, or some other failure (such as power loss or machine check) causes host processing to be abnormally terminated.

A noncatastrophic failure is one in which a particular connection is interrupted because of some malfunction in the network. Both CICS and the logical unit remain operational, but cannot communicate with each other because of the failure. In this case, the CICS node abnormal condition program (DFHZNAC) is invoked to terminate the task.

The primary objective of the message recovery procedure is to see if a message that was in-flight when a failure occurred was delivered to its destination.

Following a catastrophic system failure, any of the following CICS facilities and techniques may be used for message recovery:

- The protected task and the deferred write, which govern the logging and response activities during normal transaction processing
- The system log, which enables CICS to reconstruct the environment for any connection (represented by a TCTTE) that had a terminal message in-flight at the time of a failure

- The temporary storage message cache, which contains information related to the failing task.

These facilities and techniques are discussed in the following paragraphs. See the *CICS/MVS Recovery and Restart Guide* for further information about recovery, restart, and emergency restart.

Catastrophic failures

During a catastrophic failure, CICS does not have an opportunity to record any information concerning messages that are in-flight at the time of the failure. Therefore, selected parts of the message traffic must be recorded on the CICS system log during normal operation, so that message recovery can be performed during emergency restart. They are:

- Outbound messages that precede a synchronization point or the detachment of a task
- The initial input for a task
- Any input that follows a synchronization point.

Information concerning a message is recorded on the system log only for messages associated with a task that is protected.

Before describing the recovery techniques employed by CICS, you should understand the concept of the protected task. This concept is relevant to your decisions concerning the message option group that you must specify if you want to achieve a controlled and predictable situation for the message traffic. The message option group described below is a program control table (PCT) function that allows you to specify the CICS support necessary for message protection.

Protected tasks

If a system failure occurs and CICS emergency restart is necessary, message recovery is possible only if the task in-flight at the time of the failure was protected. CICS keeps a system log of messages preceding a synchronization point or task detach only for tasks that are declared as protected. During recovery, output messages will not be retransmitted for certain logical units, if those logical units do not support the set and test sequence number (STSN) command. You can retransmit messages using information in the temporary storage message cache (DFHM "termid").

The process of logging information for protection against a catastrophic system failure imposes an additional overhead on tasks running under CICS. Therefore, you are allowed to specify which transactions are protected and which are not. Recovery of messages after a catastrophic failure requires CICS journaling and the system log.

Nonprotected tasks should include:

- Tasks that do no more than inquire about a database
- Tasks that, when they reenter the system after a failure, are not adversely affected by double processing, if it occurs.

Double processing may occur when a task completes before the failure, but is unable to issue a completion message to the logical unit. If a task performs only an inquiry of some database record, reentering the request to recreate the reply after a system failure does not lead to invalid results. If, however, a task performs some update to a database record, and the operator does not know whether processing was complete at the time of failure, reentering the original request may mean that the database record is update twice, thus leading to erroneous results. This would not happen if the task were specified as protected.

Message option groups

To control the message protection processing for a task executing on an ACF/VTAM-supported TCTTE, you may generate message option groups, which specify the manner in which CICS DFHZCP is to treat the logical unit I/O requests for protection and recovery purposes.

You can use CEDA or DFHPCT instructions. There is no OPTGRP option in CEDA. The other options are similar to those for DFHPCT.

Using CEDA

You can use the following CEDA command to set up message protection:

```
CEDA DEFINE PROFILE(aaaa) GROUP(bbbb) MSGINTEG(YES)
 PROTECT(YES) ONEWTE(YES)
```

Then you must define your transaction:

```
CEDA DEFINE TRANSACTION(cccc) GROUP(bbbb) PROFILE(aaaa)
```

and install the group:

```
CEDA INSTALL GROUP(bbbb)
```

- Use the MSGINTEG(YES) option if a definite response is to be requested with an output request to a logical unit. Do not use it for a pipeline transaction.
- Use the ONEWTE(YES) option if the transaction is only permitted one write operation or one EXEC CICS SEND during its execution. You must also use the CEDA DEFINE TYPETERM BRACKET(YES) command or the BRACKET=YES operand of the DFHTCT TYPE=TERMINAL macro for logical units. You must use ONEWTE(YES) for pipeline transactions.
- The PROTECT(YES) option performs the same function as the macro PROTECT option described below.

You will find full information about this command in the *CICS/MVS Resource Definition (Online)* manual.

Using DFHPCT TYPE=OPTGRP

The message option group definitions should immediately follow the DFHPCT TYPE=INITIAL macro instruction. The message option group name specified by the OPTGRP operand determines which option group is to be used for the task whose PCT entry references this group. This name also must appear as a symbol prefixed to the DFHPCT TYPE=OPTGRP macro instruction, whose operands MSGPOPT and MSGPREQ specify the desired characteristics.

You specify any of the available operands as either required or optional. The MSGPREQ operand defines the processing options and characteristics that are required for the task. All of the parameters specified with MSGPREQ must be supported by the TCTTE on which the task executes. Otherwise, the task initiation request is rejected. Alternatively, the MSGPOPT operand defines the processing options that the task uses only if the TCTTE on which it is running supports the function. If the function is not supported by the TCTTE, the task initiation request is not rejected, because the functions are optional by definition. There is no default value for either MSGPREQ or MSGPOPT. If neither is specified, no options are generated for the task.

The following message group options can be specified using either the MSGPREQ operand or the MSGPOPT operand. You will find full details in the *CICS/MVS Resource Definition (Macro)* manual.

- PROTECT specifies that the task is protected, and implicitly specifies the MSGINTEG parameter. CICS also logs messages for protected tasks. PROTECT causes any write operation to a logical unit performed by a transaction to be deferred either until the CICS application program issues a terminal wait request, or until the task goes through a syncpoint or detaches. It ensures that the last message from a transaction (which confirms to the terminal operator that processing completed) is not delivered until the task has passed the commit point and cannot be backed out if a system failure occurs.
- MSGINTEG specifies that any output sent by CICS to a logical unit on behalf of a task is sent with definite response protocol specified.
- ONEWTE specifies that the transaction can perform only one write during its execution. DFHZCP sets an end bracket (EB) indicator on the first write processed for the task. Any subsequent write from the task is treated as an error by DFHZCP (because it would violate the bracket protocol), and the task is abnormally terminated. This parameter shortens the response time for simple, one-write transactions that run on TCTTEs that support the bracket protocol.

  Any of the above parameters can be specified singly or with other parameters.
- CCONTRL specifies that the application program may control the outbound chaining of request units. If you specify this option, you must not specify the MSGPREQ/MSGPOPT option PROTECT. If you specify CCONTR=YES, ONEWTE means one chain (a chain is defined as the smallest recoverable unit), and not one DFHTC TYPE=WRITE.

You can specify optional and required task options when generating the program control table (PCT). At attach time, those options that are specified as optional execute if the logical unit permits them; however, if the logical unit does not permit an optional function, the task is attached, but that option is not performed. Those options that are specified as required execute if the logical unit permits them; however, if the logical unit does not permit a required function, the task is not attached.

Emergency restart and message logging

Normally, the first entry on the log for a given protected task consists of a start-of-task indicator, the data sent by the originator of the task, and the inbound and outbound sequence numbers. If any change has been made to a protected resource (for example, a database record), the original copy is also logged.

After the task-originating data and start-of-task indicator have been logged, the task is considered in-flight. Any tasks in-flight at the time of failure are backed out by CICS during emergency restart. Backing out means that all effects of the task are removed from the system, removal being based on the records written to the system log. Naturally, messages already received or transmitted cannot be backed out, but operations (such as updates on protected system resources) that occurred during the course of the transaction can be backed out.

The end of a protected task causes CICS to write the response message, the most recent inbound and outbound sequence numbers, and the end-of-task indicator to the system log. As soon as the end-of-task indicator has been logged, the task is no longer considered in-flight; it is now considered committed and cannot be backed out.

At this point, the deferred write becomes important. If the reply to the input message had been sent as soon as it was requested (that is, before the information was logged), a period of uncertainty would exist during which the task could still be backed out, because, as mentioned previously, a task is not immune from backout until the end of the task has been logged. Because ACF/VTAM and the ACF/NCP/VS are able to operate asynchronously from CICS, the message could conceivably be delivered while CICS is abnormally terminating. In this case, the task is backed out when emergency restart is performed, yet the terminal operator has already received a message confirming that the task completed. Because backing out the task removes the effects of the task from the system, the message sent to the terminal would be inconsistent with the status of the databases.

To guard against the above situation, the deferred write is employed. This ensures that any message created by a transaction that does not specify a wait is not physically transmitted to ACF/VTAM either until a terminal control wait is explicitly specified or until the task goes through syncpoint processing. It also ensures that all messages sent to the terminal are consistent with the status of the databases.

Emergency restart

During an emergency restart of CICS, the information presented to CICS concerning the system activity at the time of the failure comprises the records that appear on the system log.

Any failure that occurs prior to the logging procedure is not detected, and CICS assumes that the task-originating data was never received. This is the same as if the failure had occurred while the task-originating data was being sent from the logical unit to CICS, and was lost in the network because of the failure.

From the data recorded on the system log, CICS reconstructs the environment for any TCTTE that had either committed output pending, or a message in-flight for a protected task at the time of failure. (The system log contains information about the terminal messages of protected tasks.) The environment reconstruction includes placing transaction-related information into the temporary storage message cache.

The message cache is an area in temporary storage with the name DFHMxxxx, where xxxx is the 4-character logical-unit identification. For a task failing in-flight, the message cache contains the data from the task-originating message, plus the transaction code, task number, and message sequence number. A flag byte indicates whether the data is for an inbound or an outbound message, and if it was logged with or without an FMH. If the failure occurred after the task had passed the commit point, the message cache contains the output message rather than the input message. If neither of these two conditions is present, the message cache is not created by emergency restart.

You may wish to write an application program to investigate the contents of the message cache following a system failure. If so, you should understand the format of the contents of the message cache. A discussion of the message cache is given in the *CICS/MVS Recovery and Restart Guide*.

Noncatastrophic failures

Environment reconstruction is not necessary following a noncatastrophic failure, because CICS and the logical unit remain operational. Message recovery is performed immediately after the failure is rectified by the node abnormal condition program (DFHZNAC) and, optionally, by the installation's node error program (NEP). Thus, DFHZNAC does not use the CICS system log.

The protected task and message integrity concepts are important because DFHZNAC uses the task's TIOA (which contains the data) in its recovery procedures. When errors are detected for tasks with message integrity, the TIOA is guaranteed to be available. For tasks without message integrity, the TIOA may have been released before the error was reported for processing. The actions taken by DFHZNAC for failing messages depend on the nature of the error and its causes; however, reliable information about a failing task is guaranteed only if the task has message integrity.

Logical unit I/O error handling (DFHZNAC/DFHZNEP)

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 the terminal abnormal condition program (DFHTACP) is scheduled to resolve terminal errors. However, there is a difference between the DFHTACP/DFHTEP interface under BTAM and the functions of the ACF/VTAM equivalents, DFHZNAC and DFHZNEP.

The implementation of terminal error processing for BTAM-supported terminals is such that any error is normally routed to the terminal abnormal condition program (DFHTACP). Depending on the type of error, DFHTACP issues messages, sets error flags, and hands over control to the user-written terminal error program, DFHTEP, a dummy version of which is supplied by CICS. After any necessary action by DFHTEP, control is handed back to DFHTACP via a DFHPC RETURN request. The interface between the node abnormal condition program (DFHZNAC) and node error program (DFHZNEP) is basically the same as that between DFHTACP and DFHTEP. You can provide, in table form, an interface module and a separate error routine for each specified transaction class. The function of the interface module, DFHZNEP, is to allow a particular transaction to have its own error processing procedure, and to determine which class of transaction is attached to the terminal, and to link from DFHZNAC to the appropriate transaction-class error routine, which is identified by a macro used in assembling DFHZNEP. On completion of the action in the transaction-class error routine, control will be returned to DFHZNAC via DFHZNEP, using the normal DFHPC RETURN request.

Note that for BTAM support, a new TACP task is created for each TACLE to be processed, whereas, for ACF/VTAM support, only one ZNAC task is created, which processes many TCTTEs.

You specify the transaction class to DFHZNEP by the NEPCLASS option of the CEDA DEFINE PROFILE command or the NEPCLAS operand of the DFHPCT TYPE=ENTRY macro. The identifier is placed in the program control table for reference by the DFHZNEPI TYPE=ENTRY macro that associates the transaction class with a named user-written transaction-class error routine. For full details on the generation and function of DFHZNEP, see the section headed "User-written node error programs" on page 133.

CICS terminal control

The terminal control macro instruction provides capabilities for CICS management modules and for customers. You can:

- Scan the terminal control table (CTYPE=LOCATE)
- Change the status of a logical unit (CTYPE=STATUS)
- Issue a ACF/VTAM indicator (CTYPE=COMMAND)
- Check the outcome of any of the above operations (CTYPE=CHECK).

For details of these facilities, see "Chapter 4.7. Modifying the terminal control table" on page 237.

| EXEC commands provide equivalent function (see see "Chapter 5.9. Examining
| and modifying resource attributes" on page 423). For upwards compatability,
| use EXEC commands rather than the macro equivalent.

Transaction options

When defining profiles used by transactions (using the CEDA DEFINE command) or transactions (using the DFHPCT TYPE=ENTRY macro instruction), you may:

- Restrict certain transactions to run only for logical units or for BTAM-supported terminals.
- Specify transaction options related to message journaling.
- Specify I/O processing options.
- Control message protection options.

The DVSUPRT operand specifies that certain transactions are permitted to execute only for a terminal or for a logical unit. The ACF/VTAM parameter of this operand allows the transaction to execute only on an ACF/VTAM TCTTE. The NONV parameter allows the transaction to execute only on a non-ACF/VTAM TCTTE (for example, a BTAM, BSAM, or a GAM TCTTE). The ALL parameter specifies that the transaction may execute on any TCTTE. ALL is the default value for the DVSUPRT operand.

The MSGJRNL operand specifies whether or not automatic journaling of messages is to be performed by the terminal control program for particular transactions. Message journaling may be requested for either input or output messages, or both.

If you specify the MSGJRNL operand, you must also specify the JFILEID operand in order to indicate where the automatic journaling information is to be recorded. The SYSTEM option indicates that the information for messages associated with logical units is to be recorded on the system log. To record the information on a particular installation journal data set instead of on the system log, specify the journal identification which can be any value in the range 2 through 99, inclusive. If you specify NO, message journaling is not performed. NO is the default value for this operand.

If you select the automatic journaling option, you must ensure that you specify the relevant journal control program and journal control table parameters to support the DFHTCP automatic journaling requests.

The I/O processing options (DELAY and IMMED) are discussed in "Output services" on page 191.

The message option groups specify the kind of protection and recovery that CICS DFHZCP will provide for the logical unit I/O messages. Message option groups were discussed under "Message option groups" on page 194.

Automatic task initiation

Before CICS attempts automatic task initiation (ATI) for a logical unit, it checks whether ATI is allowed for the particular logical unit. To permit ATI, the TCTTE that represents the logical unit must be in either the TRANSCEIVE or RECEIVE state, and must also be in service. These states are specified in the ATI(YES) option of the CEDA DEFINE TYPETERM command or the TRMSTAT operand in DFHTCT TYPE=TERMINAL.

If a logical unit TCTTE is in service but not connected when ATI is to be performed, and the CEDA option CREATESESS(YES) or TRMSTAT=INTLOG is specified, CICS requests an ACF/VTAM simulated logon to establish connection. If the CEDA option CREATESESS(NO) or TRMSTAT=NOINTLOG is specified, the ATI request is ignored until the connection is established by the other node, or the status is changed.

You must ensure that the TCTTE for which ATI is requested is either in transceive or receive state, and is in service. These parameters may be specified when the terminal control table is generated (through the DFHTCT macro instruction). They may also be specified by the ATISTATUS option of the EXEC CICS SET TERMINAL command, or by the DFHTC CTYPE=STATUS macro instruction of the terminal control interface, to enable dynamic status changes. See "SET for terminals" on page 444 and "Chapter 4.7. Modifying the terminal control table" on page 237 for further details. Where possible, use the command interface, which provides upward compatibility.

User exit routines for CICS ACF/VTAM terminal control

CICS ACF/VTAM logical unit support gives you the option of coding global user exit routines that are to be given control at defined points during the processing of a request by CICS ACF/VTAM terminal control. For more information about global user exits, see "Chapter 5.1. Global user exits" on page 289.

Because the exit routines are executed as an extension of a CICS management module, the designer of the exit routine must be fully aware of the conventions and restrictions that apply in such an environment. The exit routine must be coded in assembler language and be at least serially reusable. Requests for CICS services are forbidden in the exit routine. Issuing a wait within a management module, that is servicing a request executing under a non-user TCA can seriously degrade system performance. Furthermore, unexpected task switches from within management modules may lead to unpredictable system damage.

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

Exit	Module	Processing State
XZCATT	DFHZCP	Before a task attach.
XZCOUT	DFHZCB	Before issuing the logical message in the DFHZCP send subroutine; no chaining requirements have yet been determined.
XZCOUT1	DFHZCB	Before the message is broken into request units (RUs).
XZCIN	DFHZCB	After the entire logical message is received by CICS.

Note: XZCATT is the only exit available for LU6.2 sessions.

For XZCATT, register 8 points to the TIOA containing the transaction-originating data, and register 2 addresses a parameter list which is described in "Chapter 5.1. Global user exits" on page 289. If LU6.2 is in use, the LU6.2 transaction program name (TPN) may be accessed via the parameter list; if LU6.1 is in use, the process name (DPN) may be accessed. For all LUs, this list also contains the TRANSID of the task that CICS is about to attach. The exit can be used to specify a different TRANSID if required.

For XZCIN, if the message is too large to fit into the CICS I/O area, CICS will issue an additional receive-specific to obtain the remainder of the message from ACF/VTAM, (see "Input services" on page 190, earlier in this chapter, for further information), but control is given to the user exit routine only after the complete message has been received by CICS from ACF/VTAM.

XZCOUT is driven every time a VTAM SEND request is issued by CICS. To transmit a large message, you code a single CICS SEND command. From this, CICS can generate many VTAM SENDs. This means that an exit routine driven by XZCOUT can be invoked many times for a single message.

Alternatively, you can use XZCOUT1 to gain access to the whole message so that you can edit it before it is broken into RUs. XZCOUT1 is particularly useful when sending a large amount of data in a single message. For example, if you need to translate the data to uppercase before sending it, you need to invoke the translation routine once only, rather than once for each RU. Similarly, if your exit routine compresses VTAM data before transmission, the exit allows it to compress all of the data at once. As well as improving efficiency by calling the exit routine once only, this reduces the amount of data to be packed into the message chain, and can reduce the number of RUs in the chain.

BMS services

The level of BMS support required is specified using the BMSFUNC keyword of the DFHSG PROGRAM=BMS macro instruction, or the BMS keyword of the DFHSIT macro. Three pregenerated versions of BMS are provided. MINIMUM provides support for 3270 devices and SCS printers. STANDARD or FULL are required if you need support for any other devices. For more information on BMS support, see "BMS -- basic mapping support program" on page 14.

The batch data interchange program (DFHSG PROGRAM=DIP) must be generated if BMS routines are required for batch logical units.

Mapping individual records and entire chains

In order to map each card or line of a request unit (RU) separately, you should specify logical record presentation for the transaction (in the LOGREC operand of CEDA DEFINE PROFILE, and of DFHPCT TYPE=ENTRY) logical units for which logical record presentation and chain assembly apply. Otherwise, all records after the first in the RU will be bypassed.

In order to map an entire chain, which may consist of more than one RU, you should specify the BUILDCHAIN option of the CEDA DEFINE TYPETERM command or the CHNASSY operand in the DFHTCT TYPE=TERMINAL macro for the logical unit.

Statistics

Existing statistics are maintained for each logical unit. The following statistics are increased by one whenever the indicated condition occurs:

Statistic	Condition
Write count	ACF/VTAM SEND is accepted by a logical unit on behalf of a terminal for part of a chained output data message.
Read count	ACF/VTAM RECEIVE is completed for an input data request sent by a logical unit on behalf of a terminal. If more than one RECEIVE was necessary for CICS to obtain the complete request unit from ACF/VTAM, the read count is still increased only by one.
Error count	ACF/VTAM SEND is rejected by a logical unit on behalf of a terminal for any part of an output data message.
Error count	An exception response of any kind is received by CICS.

Statistics are kept for evaluating the size of the receive-any RPL pool. (See "Input services" on page 190, earlier in this chapter, for information about the RPL pool.) Every time DFHZCP is dispatched, it scans the pool of RPLs and counts the number of RPLs that were posted complete. DFHZCP records the maximum value of this count and increases a second counter each time this maximum is reached; every time a new maximum is recorded, the second counter is reset to one. This statistic is printed with any request that produces the existing terminal statistics; it gives the maximum value achieved and the number of times it was reached.

In a system in which the maximum value is less than the size of the RPL pool during a normal day, the number of RPLs specified for the pool could be reduced to the maximum value with no effect on system performance. Conversely, if the maximum value reaches the size of the RPL pool many times during the day, this may indicate a constraint in the system that might be causing unnecessary use of the pageable buffer area by ACF/VTAM. This situation could be improved by increasing the RPL pool size.

A good trial value for the size of the RPL pool is the maximum task value that is specified by the MXT operand of the DFHSIT TYPE=CSECT macro instruction. The value should then be reduced in accordance with the statistics recorded for peak activity. Too high a value may result in unnecessary page faulting within the RPL pool.

Another statistic keeps a count of the number of times that ACF/VTAM temporarily rejects a CICS request because there is a short-on-storage condition in ACF/VTAM. This helps you to monitor any system restraint that may arise because insufficient buffer space was allocated during ACF/VTAM definition.

Message switching

When a terminal list table is built for use with message switching, each entry in the table contains logical unit identifications. The TRMIDNT operand of the DFHTLT TYPE=ENTRY macro instruction is used to specify the identification of the logical unit to be used to direct the message.

Chapter 4.2. The CICS/TCAM interface

This chapter describes the use of TCAM under CICS/MVS 2.1.2. The following topics are discussed:

- The use of TCAM in an SNA network, with reference to protocol management, FMH processing, and error processing.
- The TCAM application program interface, including information on the process control block and the TPROCESS control block.
- The interface between TCAM and CICS, including information on terminal entries (TCTTEs) and line entries (TCTLEs) data flow, logic flow, the terminal error program, message routing, pooling, and segment processing.
- Device considerations, dealing with message formats for devices (in particular, the 3270-system devices) being used on a TCAM line.
- User exits, giving information on the three TCAM exits that may be specified in the terminal control program.
- The processes of starting up, restarting after an abend, and terminating TCAM under CICS.
- The TCAM message control program (MCP) and its relationship to the application program (in this case, CICS).

In addition, two sample TCAM message control programs for use in an SNA network are described in Appendix B, "Sample TCAM SNA message control programs" on page 515.

The majority of independent teleprocessing applications require a dedicated network. The telecommunications access method (TCAM) permits multiple applications to share a single network, resulting in more efficient use of terminals and lines. The CICS/TCAM interface enables CICS to run as an application program under TCAM.

TCAM is an access method that may be used alone or in combination with other access methods (BTAM, BSAM, ACF/VTAM, and BGAM).

One practical use of the CICS/TCAM interface is to run a production CICS system in one region and a test CICS system in another. If they run in separate regions, the applications are protected from one another. Operating under TCAM, terminals and lines can be shared by the two CICS applications. Other TCAM applications, such as the time sharing option (TSO), can also run concurrently.

CICS user tasks that run under BTAM can, in general, run under TCAM without modification to the task code. This assumes that you have correctly designed and coded the TCAM message control program. However, in order to obtain the benefits of TCAM SNA and to maintain an acceptable operator interface, it is usually necessary to change the CICS application programs to use DFHTC CONVERSE and WRITE, LAST facilities so that the MCP is provided with sufficient information about the transaction to maintain the optimum SNA message flows.

There are basic differences between TCAM and BTAM design methods. CICS was designed to operate in the BTAM environment. The CICS/TCAM interface, although resolving most of the differences, must impose some restrictions when CICS is run in a TCAM environment. These restrictions as well as some of the consequences of selecting various user options are described in this section. Also described are the user facilities available and how you implement and operate the system using the interface.

CICS with TCAM SNA

TCAM can be used to provide an SNA network without the use of ACF/VTAM. The CICS/TCAM interface has an enhanced data stream support that enables an appropriate TCAM message control program (MCP) to control the SNA session. The TCAMFET=SNA operand in DFHTCT TYPE=LINE allows TCTTEs to be specified for SNA devices. You must be prepared to write an appropriate TCAM SNA message control program to complement the CICS support and the SNA devices attached to the system. In order to obtain a good operator interface, the CICS application programs should be designed to inform the MCP of their intentions. Thus, it is better to design the MCP and the application programs together.

Sample TCAM SNA MCPs are provided in Appendix B, "Sample TCAM SNA message control programs" on page 515. The second sample MCP (DFHSPTM2) uses the information passed in the CCB to optimize the message flows to the actual logical unit. This represents transaction-oriented processing.

TCAM provides data stream support for SNA devices running under CICS. Both the SNA character string (SCS) and the 3270 data streams are supported.

In order to understand how CICS works with TCAM in an SNA environment, it is important to understand the TCAM SNA structure. The device message handler (DMH) is the logical unit in SNA terms. All data flow control (DFC), session startup and takedown, and response handling are provided in the DMH. There is no CICS control of these SNA functions, so the application programmer need not be concerned with them. For a more detailed discussion of the TCAM SNA functions provided, see the *OS/VS TCAM System Programmer's Guide*.

Protocol management

There may be many different protocols in an SNA network. The various protocols are established on a session basis using the bind image. You decide which protocols to use with which SNA session, and you should understand the requirements of the installation's application programs before deciding on a specific protocol.

Some of the more common of these SNA protocols are: bracket, half-duplex flip-flop (HDX-FF), and half-duplex contention (HDX-CON). The enforcement of these protocols is a function of the device message handler (DMH).

There are two methods of protocol management in a CICS/TCAM system:

- Device message handler control
- Transaction control.

These methods are discussed below.

Device message handler control

The device message handler method of protocol management is used when the transaction does not need to know which device it is communicating with. Although the communication control bytes are passed between CICS and TCAM, they are not used to control the SNA session. All the protocol control is provided in the DMH. You (the message handler writer, not the application programmer) choose the appearance at the outboard LU.

Transaction control

In the transaction control method of protocol management, the transaction controls the protocol. The SNA session should be bound with a protocol of HDX-FF with brackets when running this type of management. The second sample MCP provided in Appendix C is an example of a transaction controlled message handler (MH).

When using transaction control, the communication control byte (CCB) is used to relay information from the transaction to the DMH. For example:

- DFHTC TYPE=WRITE,LAST should be used to end a transaction. Issuing this macro causes an indicator to be set in the CCB requesting that the DMH send an end-of-bracket (EB) character.
- DFHTC TYPE=CONVERSE should be used when terminal input is required after a WRITE request. This macro causes an indicator to be set in the CCB requesting that the DMH send the CHANGE DIRECTION indicator to the device.
- DFHTC TYPE=DISCONNECT should be used to end the logical unit session. This macro causes an indicator to be set in the CCB requesting that the DMH terminate the LU-LU session (that is, issue the IEDHALT macro).

Function management header processing

The function management header (FMH) enables function management information to be directed to particular components within the logical unit. The FMH also provides a mechanism in which control information relating to the operation of those components may be passed. FMH processing is a bind time option (that is, a bind parameter is available to indicate whether an FMH may or may not appear in the LU-LU session).

CICS/TCAM SNA provides support for the logical device code (LDC), which is transmitted in the FMH to the logical unit. The LDC provides for the communication of the logical disposition of output to the logical unit. It can represent any meaning that is useful to the installation.

There are two ways that FMH handling can be provided. The first is for the transaction to provide the FMH as part of the data passed to TCAM by issuing a DFHTC TYPE=WRITE,FMH=YES macro. An indicator is set in the CCB so that the DMH can set the FMH included indicator in the request handler (RH) by using the IEDRH macro. On input, the DMH should interrogate the RH (using the IEDRH macro) to determine whether an FMH is included in the data. If the FMH indicator is set in the RH, the DMH should set the FMH indicator in the CCB relating to the transaction in which the input data contains an FMH.

A second method of FMH handling is to provide the entire function in the DMH. The DMH should remove the FMH before passing the input data to the transaction and insert the necessary FMH into the output data. In order for the DMH to build the correct FMH for output, some form of private interface must be established between you and the application programmer. For example, the first byte of data following the CCB can contain unique values that request specific FMH functions such as begin data set, erase record, and so on.

It is recommended that if FMH processing is required, the transaction (or preferably BMS) be used to provide the appropriate FMH.

Batch processing

When running a batch logical unit, you may want to consider is how to get the transaction identification to CICS on the begin data set condition. The alternative methods are discussed below.

The first method is for the DMH to recognize the begin data set condition by interrogating the FMH and by editing the transaction ID into the input data. This method is demonstrated in the sample MCPs provided in Appendix B, "Sample TCAM SNA message control programs" on page 515.

The second method of providing the transaction ID is for the DMH to concatenate the begin data set chain with the first chain of the data set, using the SETEOM macro. When you are using this method, the first chain of the data set must contain the transaction ID. Alternatively, the transaction ID could be set with the TCTTE beforehand by means of a permanent TRANSID or by using DFHPC TYPE=RETURN,TRANSID=xxx.

Error processing for batch logical units

During batch processing with a logical unit, there are certain logical errors from which the DMH cannot recover (for example, data set overflow or incorrect data set name). A transaction can be provided to handle these error conditions. If the transaction builds the data set on the TCAM queue and ends before the data set is transmitted, an error transaction should be created. The DMH should generate the appropriate error message or pass the SNA sense bytes to this error transaction, which then handles the error condition. If the transaction that builds the data set remains active throughout the transmission of that data set to the device, the transaction could be coded to recognize the error indicators passed to it from the DMH, rather than creating a separate error transaction.

Error processing

All error conditions, other than logical errors, are handled by the DMH. The *OS/VS TCAM System Programmer's Guide*, contains a discussion on the handling of the various sense codes returned by SNA devices. The transaction is not involved in error processing and recovery.

TCAM application program interface

The TCAM application program interface is a portion of the TCAM message control program (MCP). It consists of two types of control blocks, the process control block (PCB) and the TPROCESS block.

The PCB defines the application program interface of a region in the system using TCAM. Its purpose is to control communication and storage protection across region boundaries. It also defines the user-written message handler (MH) responsible for processing messages to and from the application program. Because a PCB is required for each application program running with the MCP, one PCB is also required to define the CICS application program.

The TPROCESS control block controls communication to and from the application program. A separate block is required for both input and output to the application program. A TPROCESS block is required for each input queue to CICS, and for each output queue from CICS. In CICS, there are corresponding terminal control table line entries (TCTLEs) for each input queue, and for each output queue (that is, for each TPROCESS block).

DD cards (such as those shown in Figure 16) are used to correlate the TCAM control blocks with the CICS control blocks. The CICS terminal control table contains the DCB. The DDNAME specified in the terminal control table macro instruction (DFHTCT TYPE=SDSCI,DDNAME=name) names the DD card. In the DD card, the QNAME field names the TCAM TPROCESS block.

You do not need to make any exceptions for CICS to the TCAM application program interface described above. For additional information, see the *OS/VS TCAM Application Programmer's Guide*.

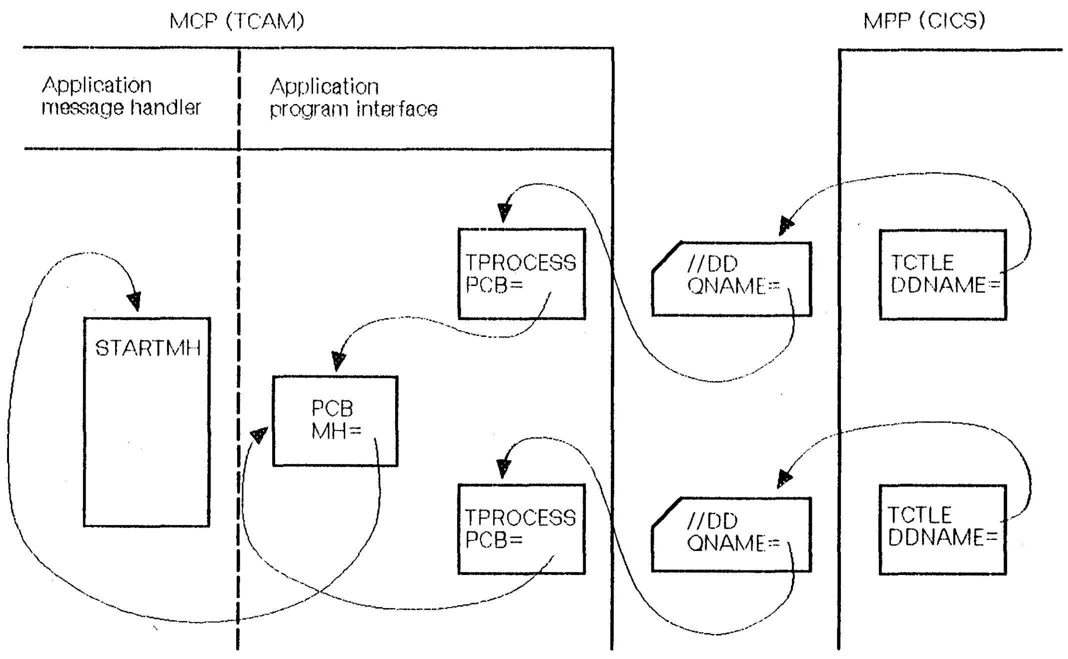


Figure 16. DD card correlation of TCAM and CICS control blocks

CICS/TCAM interface

CICS treats a TCAM input process queue as a "line." For each input process queue there is a CICS terminal control table line entry (TCTLE). Note that TCAM requires the application program (CICS) to have a DCB for each TPROCESS block; separate TPROCESS blocks are required for input to and output from the application program. Therefore, each TCAM output process queue is also treated as a line and has a corresponding CICS TCTLE. Each TCTLE references its own DCB, which is generated by the DFHTCT TYPE=SDSCI macro instruction in CICS.

The CICS terminal control table terminal entries (TCTTEs) define the terminals associated with a particular line entry (TCTLE). For each physical terminal communicating with CICS through TCAM, a corresponding TCTTE containing the terminal identification must be associated with a TCTLE. Duplicating individual TCTTEs for both the input TCTLE and the output TCTLE is avoided by attaching a single, special TCTTE to the input TCTLE and attaching all the individual TCTTEs to the output TCTLE. Although attached to the output TCTLE, they are used for both input and output processing.

Each input record from TCAM must contain the source terminal identification. Using this identification as a search argument, the corresponding TCTTE can be located by CICS by comparing against the NETNAME value for each TCTTE.

Note: The usual way of ensuring that the input records contain the source terminal identification is to specify OPTCD=W in the CICS DFHTCT TYPE=SDSCI macro. If this specification is omitted, the TCAM user is responsible for ensuring that the record contains a suitable source terminal identification.

Using the POOL=YES operand of the DFHTCT TYPE=LINE macro instruction, you can establish a pool of common TCTTEs on the output TCTLE that do not contain terminal identifiers. As required, terminal identifiers are assigned to the TCTTEs or removed from association with the TCTTEs. POOL=YES necessarily imposes a number of restrictions and should be thoroughly understood before being implemented. For additional information, see the discussion of the POOL operand in the section headed "Line pool specifications" on page 215.

Data format

When TCAM is specified, CICS assumes that the user transaction data passed to it from the TCAM queue is in the proper format to be passed directly to the user task. Except for the removal of the source terminal identification and the 2-byte CCB (if on a TCAM SNA line), CICS does not alter the data it receives. It is your responsibility (using your message control program (MCP)) to prepare the data, for example, by translating to EBCDIC, removing function management headers (FMHs), stripping line control characters, and deblocking. You may optionally bypass the CICS routine that removes the source terminal identification by returning from the user-written input exit (XTCTIN) in TCP with a displacement of 0 bytes.

Similarly, CICS assumes that the user transaction data passed to it for TCAM has been properly formatted and can be placed directly on the TCAM output process queue. Except for the insertion of the destination identification, the CCB, and the data stream control characters, CICS does not alter the data it receives. It is your responsibility (using your MCP) to prepare the data for the destination terminal, for example, by translating and inserting line control characters.

Optionally, BMS can be used with TCAM to prepare the input data for the user task, and the output data for the specific terminal type. When BMS is required with TCAM, the TRMTYPE operand in DFHTCT TYPE=LINE or in DFHTCT TYPE=TERMINAL must indicate the specific terminal type for 3270 data streams. TRMTYPE=TCAM can be used to obtain EBCDIC data stream support. For BMS support within SNA, the TCAMFET=SNA and SESTYPE operands must also be specified in DFHTCT TYPE=LINE, and in DFHTCT TYPE=TERMINAL, respectively.

Logic flow

This section gives a generalized description of the sequence of events that occurs in CICS when interfacing with TCAM. The description is in two parts, for the READ flow (input actions) and the WRITE flow (output actions) respectively.

Logic flow — READ

Figure 17 show the relationship of the fields discussed

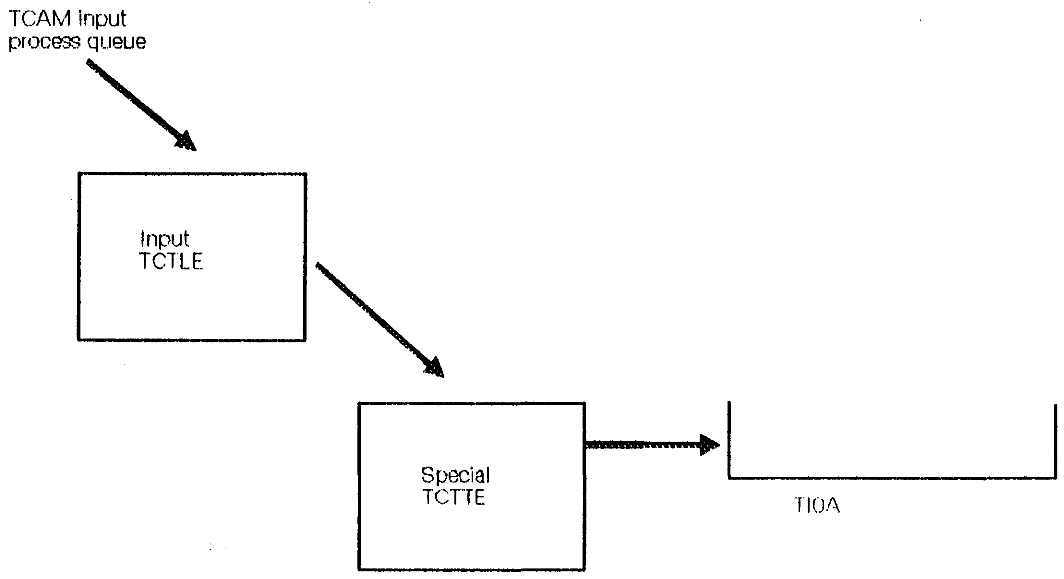


Figure 17. CICS issues a TCAM read

INPUT STEP	**ACTION**
A	TCAM notifies CICS that it has data for a particular input TCTLE by posting its ECB.
B	CICS gets a TIOA and attaches it to the special input TCTTE in the TCTLE.
C	CICS issues a READ to TCAM that results in TCAM passing the data over the region boundaries to the CICS TIOA. CICS then indicates that it has data to process. (See Figure 17 on page 211.)
D	The input TCTLE points to the corresponding output TCTLE in response to the OUTQ specification of the DFHTCT TYPE=LINE macro instruction.
E	The individual TCTTEs on the output TCTLE are searched for a matching source terminal netname. If POOL=YES has been specified, a free TCTTE is assigned to this source terminal identification. (See Figure 18 on page 213.)
F	If an input user exit (XTCTIN) has been specified, CICS links to the user exit routine where you may edit input data before passing it to a task. (See "Input user exit (XTCTIN)" on page 222.) If no exit has been specified, CICS removes the 8-byte source terminal identification field inserted by TCAM. For SNA devices, the input communication control byte (CCB) is removed. No other editing of the data is performed.
G	A check is made to determine whether a task is attached to the individual TCTTE. If there is no task attached, see Step H. If a task is attached, a check is made to see if the task has issued a READ. If a READ request exists, see Step J. If not, CICS halts the processing of data in the queue until the TCTTE is available or the attached task issues a READ.
H	CICS attaches the appropriate task. A user exit is available prior to the actual attach. (See "Task attach user exit (XTCATT)" on page 221.) If the task could not be attached (for example, if a "short on storage" or "maximum tasks" condition exists), CICS records that it has data to process, and exits from DFHTCP.
I	After a task is attached, CICS stores the TCAM segment identifier in the TCTTE (if segment processing was specified by including the C parameter in the OPTCD operand of the DFHTCT TYPE=SDSCI macro instruction).
J	CICS passes control to the attached task.

Logic flow—WRITE

Figure 18 show the relationship of the fields discussed in the list of actions that follows.

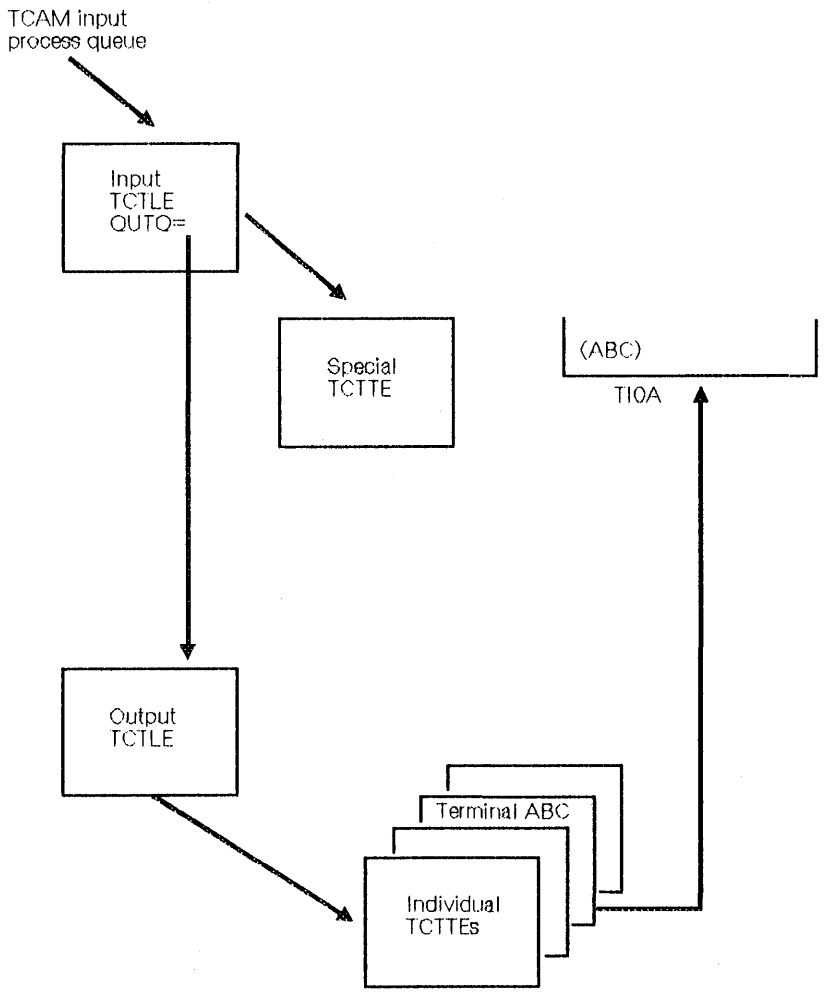


Figure 18. After TCAM read, CICS attaches TIOA to the corresponding TCTTE

OUTPUT STEP	ACTION
A	You issue a WRITE request in your application program.
B	The TCP terminal scan recognizes the WRITE request.
C	CICS checks whether an output user exit (XTCTOUT) has been specified. If it has, CICS links to the user exit routine, where you may edit your output data before passing it to TCAM. (See the discussion of XTCTOUT under "TCAM user exits" on page 221.)

D	CICS checks the four-byte TCTTE field TCTTEDES for a destination saved as a result of DEST=NAME or DEST=YES having been specified in the DFHTC TYPE=WRITE macro instruction. If it is present, CICS inserts it in the 8-byte destination field and left-justifies the field, padding blanks to the right. Otherwise, CICS moves the source terminal netname from the TCTTE to the destination field.
E	CICS moves the communication control byte (or bytes in TCAM SNA) into the 9th byte (9th and 10th bytes in TCAM SNA) of the TCAM work area. (See "TCAM devices" on page 219.)
F	CICS issues a TCAM WRITE to transfer the data to TCAM.
G	After checking for successful completion of the WRITE to TCAM, CICS flags the user task "dispatchable" if a task is still attached to the TCTTE. Otherwise, CICS frees the TCTTE for a new task.

Terminal error program

The CICS/TCAM interface implementation has resulted in the expansion of the CICS terminal error program (DFHTEP) error codes and conditions. The following errors and actions are unique to TCAM, and should be considered in DFHTEP:

Error Code	Condition	Action
X'87'(TCEMUI)	Unsolicited input	
	Terminal "receive only"	a
	Terminal "out of service"	a,b
	Task has not issued a read	No default
	No available TCTTE from pool	No default
X'9F'(TCEMIDR)	TCAM has issued an invalid destination return code to CICS.	c

where:

a = Release TCAM TIOA (X'80' at TCTLEECB+2)
b = Terminal out of service (X'08' at TCTLEECB+1)
c = Abend transaction (X'04' at TCTLEECB+1).

Message routing

The DEST operand of the DFHTC TYPE=WRITE macro instruction can be used to route an output message to a destination that you defined in the TCAM MCP. This operand can be used to send a message to a destination other than the source terminal (such as to another terminal, a list of terminals, or another application program).

If DEST=name is specified, the name is stored in the four-byte field TCTTEDES. If DEST=YES is specified, it is your responsibility to place the destination name in TCTTEDES before issuing the WRITE macro instruction.

CICS moves the data from TCTTEDES into the destination identification field before placing the data on the TCAM output process queue. You may bypass the CICS routine that inserts the destination field by taking the XTCMOUT user exit and returning to CICS from the exit with a displacement of 0. In this case, you must ensure that the TCAM header is correctly formatted for output.

If the DEST operand is omitted, CICS inserts the source terminal NETNAME from the TCTTE into the destination identification field.

Segment processing

The CICS/TCAM interface supports TCAM segment processing, except when BMS is used. It permits segments of a message to be forwarded to CICS rather than waiting for the entire message to be received. If you specify segment processing (by including the parameter "C" in the OPTCD operand of the DFHTCT TYPE=SDSCI macro instruction), CICS passes the segment to you, and places the position field control byte in the TCTTE field labeled TCTTETCM.

Similarly, on output, you must supply the control byte in TCTTETCM for CICS to pass to TCAM. If multiple terminals have been defined for one output line (that is, multiple terminals related to one TPROCESS queue), you must ensure that an entire message is passed to TCAM for a specific destination before putting the first segment for another destination on the queue. In other words, an error is returned to you if a PUT first segment to destination A is followed by a PUT first segment to destination B. For additional information on segment processing, see the discussion of the OPTCD operand of the application input and output DCB in the *OS/VS TCAM Application Programmer's Guide*.

Line pool specifications

When generating the TCAM message control program, you define each physical terminal and logical unit to TCAM by means of a TCAM TERMINAL macro instruction. Because CICS also requires terminal definitions, you must prepare a terminal control table terminal entry (TCTTE) for each terminal, or logical unit, in a DFHTCT TYPE=TERMINAL macro instruction. As a result, there is a one-to-one correlation between terminal definitions in TCAM and in CICS.

In a highly restricted environment, this duplication of terminal definitions can be reduced by using the POOL feature (DFHTCT TYPE=LINE,POOL=YES), and by specifying LASTTRM=POOL in DFHTCT TYPE=TERMINAL on the last TCTTE. Instead of a one-to-one relationship, a "pool" of generalized TCTTEs is defined for a TCAM process queue (line). When a transaction is received over the TCAM line, a search is made for an available TCTTE in the pool. When one is found, it is assigned a source terminal identification and netname for the duration of the task. When the task has completed, the TCTTE can be reassigned. If there are no available TCTTEs to handle the next transaction from the line, the line remains locked until a TCTTE becomes available because the task has ended. The number of TCTTEs in the pool influences the degree of multitasking.

You should consider providing enough terminal entries in the pool to avoid an "unsolicited input" condition (see "TCAM queues" on page 217), because there is no available entry in the pool. When DFHTCAM scans for a free entry, it will not scan the entire table because it carries a pseudo end-of-table value. Therefore, unused entries at the end of the table will never be referred to.

Line pool restrictions

You must be aware of the following line pool restrictions:

1. Because of certain device dependencies within CICS, only one terminal type is permitted for each TCAM line (process queue).
2. Automatic task initiation (ATI), transaction routing, and BMS message routing are not applicable in the pool environment. If ATI is required for certain functions and you still want to use pooling, you should consider using a special queue of nonpooled entries suitable for ATI. For example, 3270 displays could be put on a pooled entry, and 3270 printers on a nonpooled queue to cause ATI to the printers.
3. Statistics are accumulated for each TCTTE in the pool; however, the statistics cannot be correlated to the physical terminals or specific logical units.
4. Only one signon can exist for all terminal entries in a given line pool at any one time. The first signon received by CICS is communicated to all terminals in the pool. Any subsequent signon is rejected. A sign-off clears the signon data from all terminal entries in the pool; a subsequent signon is then accepted.
5. Terminal, line, and control unit requests issued by the master terminal are invalid for pooled terminals.
6. Pseudoconversational line pool implemented with an invalid RETURN transaction identification.

Line locking

Two types of line locking can occur:

1. A temporary lock that resolves itself in time, and
2. A permanent lock that remains permanent unless you take action in the terminal error program.

A temporary line lock occurs when no TCTTEs are available in the pool and a new transaction appears on the input queue. CICS locks the queue until an existing task completes execution, thus freeing a TCTTE. In this case, the completion of existing tasks is not dependent upon additional input from the queue.

A permanent line lock can occur when multiple reads are required to complete a task. For example, assume that there are two TCTTEs in the pool, that a task is attached to each, and that the messages in the input queue are in the following order:

1. Message 1 for a third transaction
2. Subsequent messages for the two active tasks.

Because no TCTTE is available in the pool for the third transaction, it must wait for a task to complete for a TCTTE to become available. Because the TCAM input queue is processed sequentially, tasks 1 and 2 are unable to receive their subsequent messages. Hence, they cannot complete, and the queue remains permanently locked.

TCAM queues

Because a queue is a sequential data set, the second message on the queue cannot be retrieved until the first message has been processed. To keep messages flowing smoothly through the queue, it is essential that each message be processed as soon as it arrives. In the CICS/TCAM interface, "processing the message" means detaching the message from the special input TCTTE and attaching it to the individual TCTTE correlated to the actual physical terminal or logical unit. Each individual TCTTE may be considered to be a "destination" for the purpose of this discussion.

If a particular destination (TCTTE) is not ready to accept the current message on the queue, the queue locks until the destination can accept the message. Queue locks are only a problem when a queue is serving more than one destination. In this case, if a queue locks, any new transaction on the queue, or messages queued for existing tasks, are not processed until the required destination has accepted the current message.

Because queue locks can adversely affect system performance, it is important that you understand their cause and effect, as described below. Proper configuration of the TCAM process queue and CICS terminal control tables reduces the occurrence and duration of queue locks to a minimum.

The maximum number of terminals that can be attached to one queue is governed by the amount of activity expected, and by the response time required from the system. For high activity and low response times, you should not exceed 25 terminals. It should be noted that only a real performance test can verify whether this figure is acceptable.

Because TCAM can read ahead from the terminals, it is possible for TCAM to present to CICS a new transaction message destined for a TCTTE that is already processing a task. Also, TCAM can present a message for an existing task before that task issues a READ request. In either case, CICS cannot process the message (as described above) until the TCTTE is ready to accept the new TIOA. Such input is called "unsolicited input."

Five conditions can produce unsolicited input:

1. The CICS TCTTE for which the data is destined is out of service.
2. The CICS special input TCTTE for the associated input queue is out of service.
3. The CICS TCTTE for which the data is destined is in RECEIVE status.
4. The CICS TCTTE for which the data is destined has an associated task that has not issued a READ, and for which the period of time indicated by the NPDELAY specification has expired.
5. A terminal in a pool has entered data and is unable to find an available TCTTE.

In all cases, the action taken by the CICS/TCAM interface is to place the input line out of service, and attach DFHTACP to process the error condition.

The default action taken by DFHTACP (which can be altered by a user-written DFHTEP) for conditions 1, 2, and 3 is to discard the data and place the input line in service. No default action is taken by DFHTACP for condition 4 or 5; therefore, the input line is placed in service, but with the same message still to be processed, thereby preventing CICS from reading any subsequent messages from the input queue.

To allow processing of input to continue, DFHTEP may take either of the following steps:

- If the input line is placed in service by DFHTEP, the CICS/TCAM interface retries the operation. In this case, a count mechanism is recommended in DFHTEP to prevent a loop occurring if the task never issues a READ, or a TCTTE never becomes available, or:
- Perhaps when a count limit is reached, DFHTEP might abend the task, dispose of the data, and place the line in service.

For further information concerning DFHTEP, see "Terminal error program" on page 72.

The problem of unsolicited input caused by condition 5 can be eliminated entirely by having a separate TCAM input process queue for each CICS terminal (TCTTE). However, as the number of terminals increases, this solution may quickly use up too much main storage.

You should analyze the type of traffic that you anticipate over the queues. If a 2770 Data Communication System, or a 2780 Data Transmission Terminal, is to read in volumes of cards, you should consider having separate queues for these devices. However, devices can share queues if traffic is conversational with short-lived tasks. The same TCAM output process queue can be specified for multiple input process queues. (See the discussion of the DFHTCT TYPE=LINE,OUTQ=symbolic name specification in the *CICS/MVS Resource Definition (Macro)* manual.)

You need not be concerned with locking of the TCAM output process queue, because TCAM requeues the data according to its final destination once it arrives over the output queue.

It is possible for the TCAM output process queue to become congested because of lack of queuing space. In this case, CICS has a WRITE to the queue outstanding until TCAM accepts the data.

If the length of a message passed to a TCAM queue exceeds the queue's block size (specified in the DCB), DFHTCAM will cause DFHTACP to be attached with "output area exceeded" (error code X'8F').

TCAM devices

In a non-TCAM environment, the CICS terminal control program is responsible for polling and addressing terminals, code translation, transaction initiation, task and line synchronization, and the line control necessary to read from or write to a terminal. When TCAM is specified, terminal control relinquishes responsibility to the TCAM MCP for polling and addressing terminals, code translation, and line control. To take advantage of TCAM facilities, code in the MCP message handler functions, such as code translation, which were previously handled by the CICS terminal control program.

For some terminal services, it is necessary for CICS to pass the user request on to the TCAM MCP message handler. A communication control byte (2 bytes in TCAM SNA) in the TCAM work area has been established for this purpose. It is passed to TCAM along with the 8-byte destination name field. You must execute the appropriate MCP message-handler macro instructions according to the contents of the communication byte.

The terminal services parameters that do not set bits in the communication control byte are WRITE, WAIT, and SAVE. Bits in the communication control byte are set for the DISCONNECT, FMH, and CONVERSE parameters, and for the LAST parameter on the WRITE macro.

The CICS/TCAM interface does not support the RESET parameter or the 3270 parameters READB and COPY.

All messages to TCAM from CICS are prefixed with the standard CICS/TCAM communication area. This is one byte for the non-SNA TCAM interface, and two for the TCAM SNA interface (that is, when TCAMFET=SNA is specified in DFHTCT TYPE=LINE). This area is used to convey to TCAM special requests and options that cannot be used within CICS.

The format of the communication area is:

First byte

FMH present in stream	X'01'
Extended CCB (2-byte CCB)	X'04'
DISCONNECT request	X'08'
READL (read keyboard)	X'10'
WRITEL (write keyboard)	X'20'

Second byte (present if extended CCB is on)

Last output from transaction	X'01' (WRITE,LAST)
READ requested after this WRITE	X'02' (WRITE,READ request or CONVERSE)

All other flags are reserved and are set to 0.

Generalized TCAM message format

Messages passed to CICS from TCAM and vice versa have the following format:

Destination	CCB	Device-dependent data	FMH	Message
8 bytes	2 bytes (optional) (SNA only)	x bytes (device-dependent)	y bytes (SNA only)	

Destination	Destination name (8 bytes) taken from TCTTE's netname parameter or from DEST specification on output.
CCB	Communication control byte(s) This determines the options specified for the message (for example, whether an FMH is present or not). The length of the CCB varies, and can be: 0 bytes (input message non-SNA) 1 byte (output message non-SNA) 2 bytes (input/output messages — SNA).
Device-dependent data	Dependent on the device — 3270, or other. See the following sections on the relevant devices.
FMH	Function management header. SNA only = length in first byte non-SNA = not applicable.
Message	User data.

TCAM with 3270 devices

The CCB and device-dependent data for an input message from TCAM to CICS have the following format:

CCB1 1 byte SNA only	CCB2 1 byte SNA only	Attention identifier 1 byte	CURSOR 2 bytes

The CCB is present for TCAM SNA lines only.

The CCB and device-dependent data for 3270 output messages from CICS to TCAM have the following format:

CCB1 1 byte	CCB2 1 byte	1 1 byte	2 1 byte	3 1 byte

```
(2 bytes - SNA) device-dependent data (1 byte - non-SNA)

                    1 Escape character 2 Command 3 WCC (write
                    control character)
```

All SOH% status messages input to CICS are passed to DFHTACP or DFHTEP.

Terminal control copy and read buffer requests are not supported by the CICS/TCAM interface.

In addition to normal read/write functions, the ERASEAUP, CTLCHAR, and UCTRAN operands are also valid for the 3270.

All 3270 printer scheduling and error handling is provided by the TCAM message handler.

The user data stream for 3270 data stream devices contains user data and control characters.

Note: For 3270 SDLC devices, the escape character must be removed by the message handler.

TCAM user exits

There are three TCAM user exits. How to use global user exits is described in "Chapter 5.1. Global user exits" on page 289.

The three user exits are XTCATT, XTCTIN, and XTCTOUT. Whereas XTCATT is shared by other users, XTCTIN and XTCTOUT are available only to TCAM users, and are in place of the XTCIN and XTCOUT exits used by others.

Task attach user exit (XTCATT)

The XTCATT exit is invoked prior to issuing a task control ATTACH for a transaction identification received in response to polling. In the CICS/TCAM interface, this information is received over the TCAM input process queue.

Input user exit (XTCTIN)

The XTCTIN exit is invoked following the completion of any input event, (that is, after the individual TCTTE has been located, but just before CICS checks to see if a task is attached to the TCTTE). At this time, the LIOA contains the 12-byte storage accounting field and the work area from TCAM. The work area contains an 8-byte source terminal identification header, the CCBs (if TCAM SNA), and the work unit (user data). TIOABAR (register 4) points to the line I/O area containing the origin field and user transaction data. TCTTEAR (register 2) points to the corresponding TCTTE for this message, and the TCTTEDA field within the TCTTE points to the TIOA that is to be used to contain the edited message.

You have two options when returning from the user exit. If a return code of 0 is issued, CICS removes the 8-byte source terminal identification field and the CCBs (if TCAM SNA input). Upon completion, the TIOA contains the 12-byte CICS storage accounting field and the work unit. (See Figure 19 on page 223.)

If you return from the exit with a return code of 4, CICS does not alter the data in the TIOA. It is then your responsibility to handle the TCAM header.

For a discussion of TCAM work areas and work units, see the *OS/VS TCAM Application Programmer's Guide*.

Output user exit (XTCTOUT)

The XTCTOUT exit is invoked for output events before placing data on the TCAM output process queue.

You have two options when returning from the exit. If you return from the exit with a return code of 0, CICS inserts in the TIOA, between the 12-byte CICS storage accounting field and the work unit, a TCAM header consisting of an 8-byte destination field and the communication control byte, or bytes, required for TCAM. If you return from the exit with a return code of 0, CICS obtains an LIOA if necessary, inserts a TCAM header consisting of an 8-byte destination name, communication control area, and any device dependent data, and copies user data from the TIOA. The LIOA is then used to transmit the data to the TCAM queue. If you return from the exit with a return code of 4, CICS bypasses this insertion routine. It is then your responsibility to ensure that the TCAM header is properly formatted.

Data flow

Figure 19 shows the composition of the TCAM work area, and the CICS line and terminal input/output areas (LIOA and TIOA) at the various stages of operation.

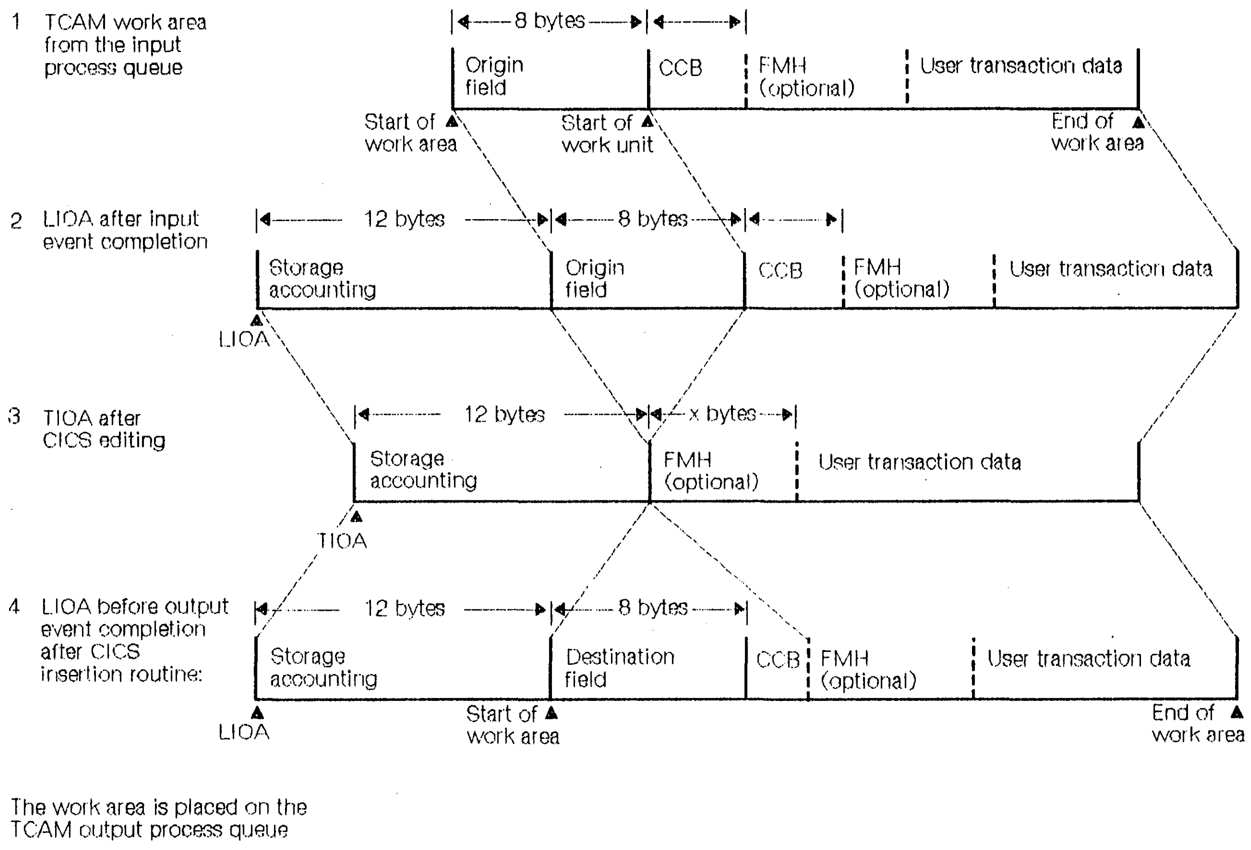


Figure 19. Stages of TCAM work area and CICS input/output areas

On input 1, note the information available from the TCAM input process queue. At 2, the CICS/TCAM interface has obtained a line I/O area, and has received the TCAM message into that area. This is the state when input event completion has just taken place. If default editing is then performed, a TIOA (as at 3) is obtained and the relevant data is copied from the LIOA in 2 to this TIOA (that is, the origin field, CCB (if any), and device dependent data are removed). This TIOA is then given to you. On output, a TIOA (as at 3) is provided by the user. The CICS/TCAM interface obtains an LIOA (at 4) if necessary, and inserts a destination name, a CCB, and device dependent data before copying the user transaction data. This information, beginning at the start of the work area, is placed in the TCAM output process queue.

The TCAM origin field contains the source terminal network name (netname).

The TCAM destination field contains the destination identification so that TCAM can route the data correctly.

If you specify the output user exit and return from the exit with a return code of 4, CICS does not alter the TIOA work area. You must provide the data length at TIOATDL and prepare the work area for TCAM, which includes the 8-byte destination field and the communication control byte (bytes if TCAM SNA).

CICS/TCAM startup

The TCAM MCP must be in operation before completing CICS system initialization. When you bring up CICS with the CICS/TCAM interface, CICS checks for the presence of a TCAM region and issues the operator message:

DFH1500 – CICS CHECKING FOR TCAM MCP.

If CICS discovers the MCP is not operational, the following messages are issued:

DFH1520 – TCAM MCP IS NOT CURRENTLY AVAILABLE
DFH1520 – REPLY RETRY OR CANCEL OR CONTINUE.

The operator must then respond:

RETRY

when the TCAM region becomes active; or

CANCEL to terminate CICS; or

CONTINUE

to continue initialization of CICS in the absence of the TCAM region.

If the operator responds CONTINUE, all DD cards that refer to a TCAM queue must have been previously removed from the startup deck to avoid an abnormal termination of CICS. The CONTINUE response is applicable to a mixed BTAM/TCAM mode of operation when TCAM lines are not being used during execution of CICS.

CICS/TCAM ABEND/RESTART

If the TCAM message control program (MCP) terminates abnormally, any TCAM application programs currently active are automatically terminated abnormally, providing there is at least one open line group in the MCP. This also applies to the CICS application program. For further information, see the relevant sections in the *OS/VS TCAM System Programmer's Guide* and in the *OS/VS TCAM Application Programmer's Guide*. CICS does not provide RESTART capability.

CICS/TCAM termination

CICS is terminated in the normal manner. No modifications to termination procedures are required to support the CICS/TCAM interface. If both CICS and TCAM are being terminated, CICS should be terminated first to avoid an abnormal termination of CICS.

CICS and TCAM: program interrelationship

Figure 20 illustrates the interrelationship between the TCAM message control program (MCP) and the TCAM application program. CICS is regarded as an application program by TCAM.

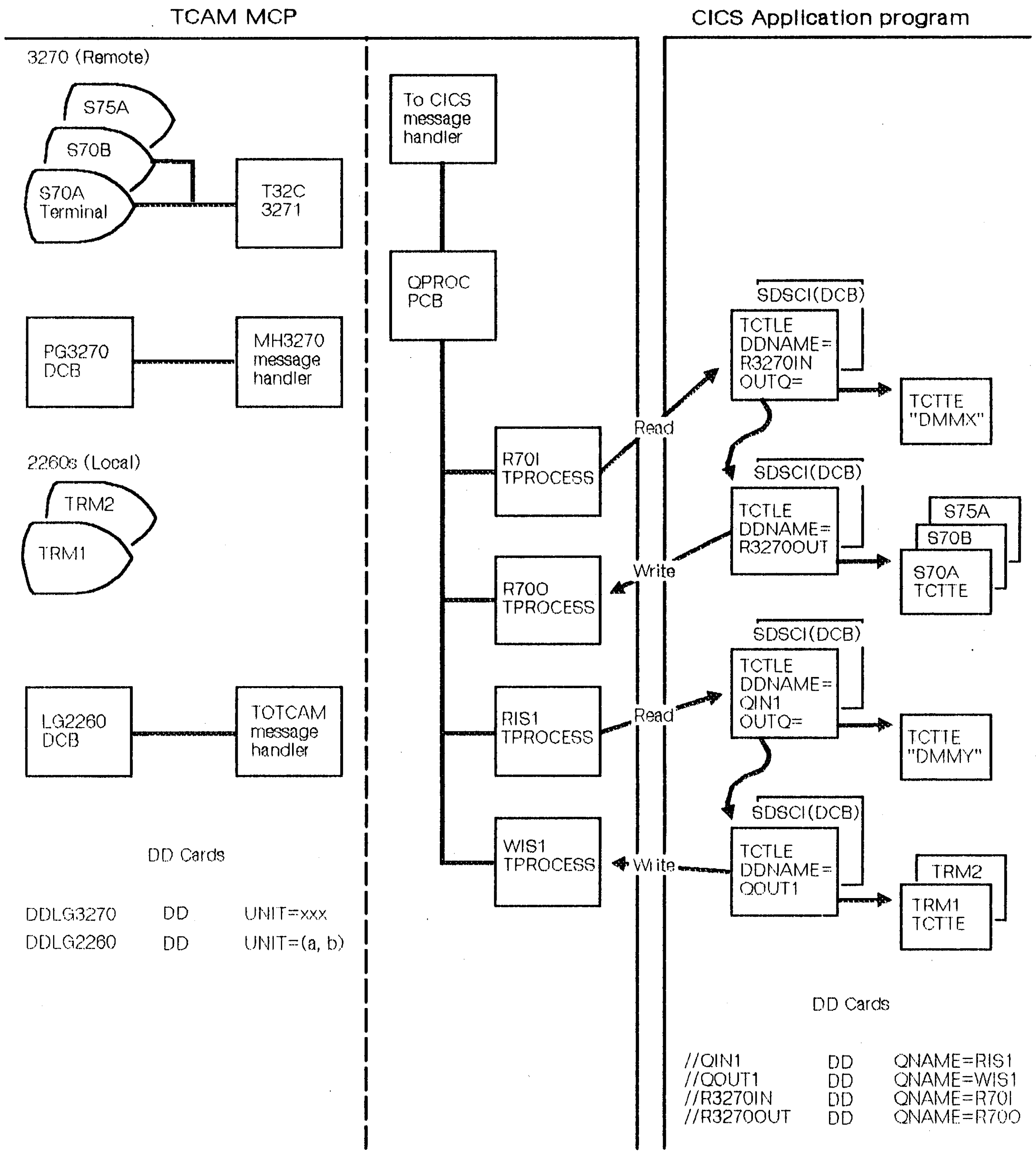


Figure 20. TCAM message control and application program

The following is an example of a TCAM message control program. This MCP shows the relationship between the MCP definition and the CICS terminal control table generation. It should not be construed as a typical or usable MCP for CICS. For further information on MCP definition for a variety of devices, see the *OS/VS TCAM Installation and Migration Guide*. An example of a CICS terminal control table for TCAM is given in the *CICS/MVS Resource Definition (Macro)* manual.

TCAM message control program (non-SNA)

```
MCPCICS    CSECT
TCAMINIT   INTRO    DISK=NO,                                              *
                    PROGID=TCAM/CICS,                                     *
                    LUNITS=40,                                            *
                    MSUNITS=20,                                           *
                    KEYLEN=132,                                           *
                    CROSSRF=4,                                            *
                    DLQ=TRM1,                                             *
                    STARTUP=CY,                                           *
                    TRACE=10,                                             *
                    LINETYP=BOTH,                                         *
                    OLTEST=0
           LTR      15,15
           BZ       OPENLINE
NOEXEC     ABEND    123,DUMP
OPENLINE   OPEN     (PG3270,(INOUT))
           TM       PG3270+48,DCBOFLGS
           BNO      NOEXEC
           WTO      'TIME TO START APPLICATION PROGRAM'
           READY
FINISH     CLOSE    (,PG3270)
           L        13,4(13)
           RETURN   (14,12)
PG3270     DCB      DSORG=TX,MACRV=(G,P),CPRI=S,DDNAME=DDPG3270,          *
                    MH=MH3270,PCI=(N,N),BUFSIZE=464,                      *
                    INVLIST=(POLL70R,,),TRANS=EBCD
QPROC      PCB      MH=TOCICS,                                            *
                    BUFSIZE=464,                                          *
                    RESERVE=(20)
           TTABLE   LAST=TRM2
RIS1       TPROCESS PCB=QPROC,QUEUES=MO                                   *
WIS1       TPROCESS PCB=QPROC
R70I       TPROCESS PCB=QPROC,QUEUES=MO
R700       TPROCESS PCB=QPROC
T32C       TERMINAL QBY=T,DCB=LG3270R,RLN=1,TERM=327C,QUEUES=MO
S70A       TERMINAL QBY=T,DCB=PG3270,RLN=1,TERM=327R,QUEUES=MO,           *
                    ADDR=6161404020,NTBLKSZ=1856,SECTERM=YES
S70B       TERMINAL QBY=T,DCB=PG3270,RLN=1,TERM=327R,QUEUES=MO,           *
                    ADDR=6161C1C12D,NTBLKSZ=1856,SECTERM=YES
S75A       TERMINAL QBY=T,DCB=PG3270,RLN=1,TERM=327R,QUEUES=MO,           *
                    ADDR=E2E240402D,NTBLKSZ=1856
POLLST1    INVLIST  ORDER=(TRM1+02)
POLLST2    INVLIST  ORDER =(TRM2+02)
```

```
POLL70R   INVLIST ORDER =(T32C-C1C17F7F2D,S70A+C1C140402D,           *
                        S70B+C1C1C1C12D,                              *
                        S75A+C2C27F7F2D,S75A+C2C240402D),             *
                        EOT=37
*
TOTCAM    STARTMH LC=OUT                  2260 MH
          INHDR                           PROCESS INCOMING MSG HEADER
          CODE
          FORWARD DEST=C'RIS1'
          INMSG                           PROCESS INCOMING COMPLETE MSG
          INEND                           END OF INCOMING SECTION
          OUTHDR                          PROCESS OUTGOING MSG HEADER
          OUTEND                          END OF OUTGOING SECTION
*
MH3270    STARTMH LC=OUT,CONV=YES,STOP=YES  3270 MH
          INHDR,                          PROCESS INCOMING MSG HEADER
          MSGTYPE X'6C'                   IS IT A STATUS MESSAGE
          B       SOH                     IF SO, BRANCH ROUND
          MSGTYPE                         ALL OTHER MSGS TO HERE
          MSGEDIT ((R,CONTRACT,SCAN,(2))),BLANK=NO  REMOVE CUDV BYTES
SOH       EQU * or DS 0H
          CODE
          FORWARD DEST=C'R70I'
          INBUF                           INCOMING MSG SEGMENT SECTION
          INMSG                           SECTION FOR COMPLETE MSGS
          INEND                           END INCOMING SECTION
          OUTHDR
          SETSCAN 1
          MSGEDIT ((R,,SCAN))             REMOVE CCB
          MSGFORM                         SETS IN LINE CONTROL CHARS
          CODE
          OUTBUF,
          OUTMSG
          OUTEND                          END OUTPUT SECTION
TOCICS    STARTMH LC=OUT                  MH FOR APPLICATION
          INHDR
          CODE                            TRANSLATE TO EBCDIC
          FORWARD DEST=PUT                WRITE TO CICS
          INEND
          OUTHDR
          OUTEND
DCBOFLGS  EQU     X'10'
          END
```

Chapter 4.3. Writing a transaction to IPL the IBM System/7

Diagnosis, Modification, and Tuning Information

The IBM System/7 may be used under two line protocols in a CICS environment. The chapter provides information on writing a transaction to initial program load (IPL) the System/7 on start/stop and on BSC lines.

On a start/stop line

To IPL the System/7 from CICS, you must write a transaction that issues an automatic transaction initiation request to either interval control or transient data control. This transaction is usually initiated from the master terminal or from a sequential terminal. The initiated transaction is started on the System/7; it then writes the IPL records to the System/7.

You prepare the IPL records, which must consist of:

- UZERO, a utility module
- UTIPL, a utility module
- System/7 storage load.

UZERO and UTIPL are provided in object deck form on the MSP/7 distribution tape under member names CAAUZERO and CAAUTIPL, respectively. If link-edited into the user-written application program, UZERO and UTIPL become available for transmission in a suitably translated format.

The first two bytes of each of these modules contain a count of the number of characters in the remainder of the module. These two bytes must be placed in your TIOA at TIOATDL by the application program. The remainder of the module is moved to TIOADBA. UZERO and UTIPL may then be transmitted to the System/7 by issuing terminal output requests with the WAIT option in the application program.

The System/7 storage load is generated by the formatting utility (FORMAT/7) by specifying "PARM=TCOM" in the execute card of the formatting job step. The storage load is comprised of 80-character records that may be read using the transient data or file control facilities of CICS and transmitted to the System/7 by issuing a series of terminal output operations with the WAIT option. If a DFHPC RETURN request is used to allow the System/7 to begin execution, you must ensure that no automatically initiated transaction is scheduled to begin on the System/7 until at least 10 seconds have elapsed following execution of the DFHPC RETURN request.

For more information concerning the preparation of IPL records for the System/7, see the publication *IBM System/7 MSP/7 Host Program Preparation Facilities II on System/360* or System/370: Assembler, Source Preparation Program and Formatting Utility*, GC34-0018.

Using a BSC line

CICS supports the IPL of a System/7 with the binary synchronous communications adapter (BSCA) using a multipoint line only. This feature requires that a terminal entry (TCTTE) be generated which includes the following parameters:

```
TRMTYPE=S/7BSCA,
TRMSTAT=IPL,
TRMADDR=label,
FEATURE=TRANSPARENCY,...
```

The DFTRMLST pointed to by the TRMADDR parameter must specify an address in the form (SEL SEL DC1 DC1 ENQ), where SEL is the System/7 selection address. This logical terminal is used exclusively for the IPL of the System/7. One additional TCTTE is required for each logical terminal in the System/7. The number of logical terminals that reside in a System/7 is limited by the application program running in the System/7.

No entry should be made in the polling list for the System/7 IPL logical terminal.

To IPL the System/7 from CICS, you must write a transaction that issues an automatic transaction initiation request to either interval control or transient data control. This transaction is usually initiated from the master terminal or from a sequential terminal. The initiated transaction is started on the System/7; it then writes the IPL records to the System/7.

You prepare the IPL records, which consist of the following:

- $UBIPL (the bootstrap loader)
- System/7 storage load.

$UBIPL is supplied with MSP/7. You write and assemble the System/7 storage load. You must specify CARD format for the execution of FORMAT/7, the MSP/7 formatting utility. The user-written CICS transaction that transmits the $UBIPL and the storage load records to the System/7 uses the following macro:

```
DFHTC TYPE=(WRITE,WAIT,TRANSPARENT)
```

For further information, see *MSP/7 Macro Library/Relocatable: Coding the Input/Output Macros*, GC34-0020.

* IBM Trademark. For a list of trademarks see page iii.

Chapter 4.4. IBM 3735 Programmable Buffered Terminal

This chapter provides a summary of the specific options that must be included in the CICS system generation and table preparation macro instructions to provide support for the IBM 3735 Programmable Buffered Terminal in a switched line network. The 3735 inquiry mode feature is also discussed.

System generation

BTAMDEV=3735D and ANSWRBK=EXIDVER must be included in the DFHSG PROGRAM=TCP macro instruction during system generation.

Terminal control table preparation

FEATURE=AUTOANSR must be specified in the DFHTCT TYPE=LINE macro instruction for all terminals on switched-line networks. To support the 3735 Programmable Buffered Terminal, the following must also be specified:

- DFHTCT TYPE=LINE,ANSWRBK=EXIDVER.
- BTAM DFTRMLST macro instruction of the form SWLST,AN. The user portion of each 3735 DFTRMLST entry must point to the corresponding TCTTE.
- DFHTCT TYPE=TERMINAL,TRMTYPE=3735.

If FEATURE=AUTOCALL is specified in the DFHTCT TYPE=LINE macro instruction, the following must also be specified:

- BTAM DFTRMLST macro instruction of the form SWLST,AD.
- DFHTCT TYPE=TERMINAL,TRMADDR=parameter.

The TRANSID operand is required for batch input in the form TRANSID=xxxx where xxxx is the transaction identification of the user-written batch processor.

Inquiry mode

CICS deletes the inquiry header on input and inserts it on output. Therefore, inquiry applications require that:

- Only one output record is transmitted.
- The output block does not exceed 233 bytes (plus a three-byte inquiry header).
- The output data stream does not contain characters that are invalid for a 3735. (See the *IBM 3735 Programmer's Guide*, GC30-3001.)

If multiple inquiries are required in a single connection on a switched line, you must make provision in the DFHTEP program to keep the line open. To accomplish this, you may check for the IOERROR — TIMEOUT condition, a WRITE TR or READ TQ instruction, and the contents of TCTTEMCI for the value TCTTEMIQ, which is a hexadecimal blank character (X'40').

Chapter 4.5. IBM 3740 Data Entry System

This chapter contains information on the macros and operands that must be specified during the CICS system generation and table preparation process that provides support for the IBM 3740 Data Entry System in a switched line network. The 3740 expanded ID verification feature is also discussed.

System generation

BTAMDEV=3740D must be included in the DFHSG PROGRAM=TCP macro instruction during system generation.

Sign-on table preparation

For each operator identification card that is used to sign on using the 3741, the following parameters must be specified in DFHSNT TYPE=ENTRY:

- OPIDENT=ccc
- OPNAME=dd...d
- NAMFORM=DEC

Terminal control table preparation

FEATURE=AUTOANSR must be specified in the DFHTCT TYPE=LINE macro instruction for all terminals on switched-line networks. To support the 3740 Data Entry System, the following must be specified:

- BTAM DFTRMLST macro instruction of the form SWLST,AN. The user portion of each 3740 DFTRMLST entry must point to the corresponding TCTTE.
- DFHTCT TYPE=TERMINAL,TRMTYPE=3740.

If FEATURE=AUTOCALL is specified in the DFHTCT TYPE=LINE macro instruction, the following must also be specified:

- BTAM DFTRMLST macro instruction of the form SWLST,AD.
- DFHTCT TYPE=TERMINAL,TRMADDR=parameter.

ID verification

If the 3740 does not have the expanded ID verification feature (specified in the ANSWRBK=EXIDVER operand of DFHTCT TYPE=LINE macro), the first record (block) from the 3740 must contain only the terminal identification; any other data in the first block will be disregarded. Data must begin in byte 1 of the second block.

Chapter 4.6. IBM 3600 Finance Communication System in a BSC network

This chapter contains information on the CICS system generation and resource definition options needed to support the IBM 3600 Finance Communication System in a BSC network. It also describes the 3600 buffer depletion feature.

System generation

BTAMDEV=3600 must be specified in DFHSG PROGRAM=TCP to generate 3600 BSC support. Other terminal control program parameters apply as follows:

- FEATURE=TRANSPARENCY must be specified if CICS and 3601 application programs are to communicate in transparent mode.

  DUALADR=YES must be specified for all logical work stations attached to a controller which performs dual address character insertion in response to a general poll.
- BSCODE=EBCDIC is required for 3600 BSC support.
- FEATURE=AUTOPOLL is required.
- WRAPLST=YES should only be specified if the wrap list feature is to be included in CICS.

Terminal control table preparation

The following must be specified in DFHTCT TYPE=SDSCI for 3600 BSC devices:

- DEVICE=3600 if all terminals in the line group are 3600s or DEVICE=BSCMDMPT for mixed binary synchronous multipoint devices present in the line group.
- BSCODE=EBCDIC.

The poll list generated by the DFTRMLST macro must conform to the general poll requirements described for BAM in *IBM 3600 Finance Communication System: Customer Feature Description for BSC3 Communication*, GC22-9026. CICS support requires that a 1-character component address be specified in the 3601 CPGEN as the poll address. If necessary, the 3600 entries must be padded with leading SYN characters if the line to which the 3600 devices are attached also contains other device types, because the poll list entries must all be of the same length.

The following must be specified in the DFHTCT TYPE=LINE macro:

- TRMTYPE=3600. If a remote 3270 and a 3600 BSC device are both on one line, TRMTYPE must specify the remote 3270.
- GENPOLL=YES. This is the default when TRMTYPE=3600, 3270, or 2980.
- BSCODE=EBCDIC. This is the default.

- INAREAL must accommodate the maximum input length, including data link control characters, from any device on the line. If a remote 3270 is attached to the line, the length must not be less than 254. For 3600 control units sending unblocked data, the length must not be less than the largest message segment written to the host by any single work station. For 3600 control units sending blocked data, the length must accommodate the maximum allowable transmission, as specified in the 3600 CPGEN.

The following relate to the DFHTCT TYPE=TERMINAL macro:

- TRMTYPE=3600 indicates a 3600 BSC device when the SDSCI and LINE macros have also been specified thus. Otherwise, ACF/VTAM 3600 support will be generated.
- FEATURE=TRANSPARENCY must be specified if the CICS and 3601 application programs issue transparent writes.

If BUFFER=0 is specified or defaulted, CICS sends output to the 3601 in one transmission without segmenting it. Thus, both the 3601 host input buffers and the receiving work station's host input segment must be large enough to accommodate any CICS application program or system message that can be sent to the work station.

BMS parameters must not be specified, as BMS does not support 3600 BSC devices.

Buffer depletion

Buffer depletion occurs when the CICS terminal control program attempts to send a message segment to a 3600 controller and receives an indication that the 3600 has no buffers currently available to receive data from the host. Each data transmission from CICS occupies a 3600 controller buffer until a work station reads the data into its work area. Thus, buffer depletion may occur when 3600 work stations are not reading data sent by the host. If the CICS terminal control program detects a buffer depletion condition, it waits 1.5 seconds and then retransmits the segment. This sequence is repeated until the 3601 has a buffer available to receive the segment, or until some other error occurs.

End of Diagnosis, Modification, and Tuning Information

Chapter 4.7. Modifying the terminal control table

Important

There are two ways to modify the terminal control table (TCT) dynamically:

Using the macros described in this chapter

Using the commands described in "Chapter 5.9. Examining and modifying resource attributes" on page 423.

The use of commands is recommended because this provides upwards compatibility with later CICS releases and because commands can be issued from all supported languages.

This chapter provides reference information on the macro and operands of the terminal control macro instruction interface (DFHTC CTYPE macros). The functions and relevant macro instructions of this interface are:

- Scanning the terminal control table (DFHTC CTYPE=LOCATE)
- Changing the status of a logical unit (DFHTC CTYPE=STATUS)
- Checking the outcome of any of the above operations (DFHTC CTYPE=CHECK)
- Issuing an ACF/VTAM indicator (DFHTC CTYPE=COMMAND).

If an address is returned after a DFHTC CTYPE request, it should be assumed to be valid only until the next CICS request is issued. CICS reserves the right to reposition internal control blocks during the execution of a transaction. Therefore, after each CICS request that causes a CICS wait, a new DFHTC CTYPE request should be issued to readdress the control block.

You should use the DFHTC CTYPE macros only when writing user-specific routines to handle recovery and error correction conditions.

These macros are only available for use with the macro-level application programming interface, and only with assembler language.

There is a description of the DFHTC CTYPE macros and operands below.

Notes:

1. You must specify DFHTCTZE CICSYST=YES and DFHTCA CICSYST=YES in order to generate the system portions of the TCTTE and TCA DSECTs, which are required for any program that uses the DFHTC CTYPE requests and commands.
2. Field TCATPTA may be an input and an output field. A value placed in TCATPTA will be overwritten by the answer, for example, if you are using the TERM=ID operand.
3. At the command level, EXEC CICS INQUIRE and SET commands may be used to perform similar functions to those described in this chapter. You should look at "Chapter 5.9. Examining and modifying resource attributes" on page 423 to see if you can use command level rather than macro level.

Terminal locate function — DFHTC CTYPE=LOCATE

You may use the DFHTC CTYPE=LOCATE macro instruction to:

- Find the TCTTE for a local or remote terminal, or a session.
- Find the TCTSE (system entry) for a route to a CICS region in the network.
- Retrieve LDC information associated with a TCTTE.
- Scan the TCT from top to bottom.

The locate function allows you to perform any of the above operations without being concerned with the structure of the terminal control table. For example, you can use the function to keep track of the availability of certain printers to schedule output to them, instead of implementing table-dependent application programs to do so.

```
DFHTC   CTYPE=LOCATE
        [,DOMAIN={LOCAL|REMOTE|ALL
                 |SYSTEM|SESSIONS}]
        [,ERROR=symbolic-address]
        [,INVADDR=symbolic-address]
        [,INVID=symbolic-address]
        [,LASTTRM=symbolic-address]
        [,LDC={DEFAULT|YES}]
        [,NETNAME={FIRST|NEXT|ID}]
        [,NORESP=symbolic-address]
        [,SELECT=([INTLOG][,NOINTLOG][,INSRV]
                 [,OUTSRV][,VTAM]
                 [,PRIMARY][,SECONDARY]
                 [,RELEASED][,ACQUIRED])]
        [,STATUS=([INTLOG][,NOINTLOG][,INSRV]
                 [,OUTSRV][,VTAM]
                 [,PRIMARY][,SECONDARY]
                 [,RELEASED][,ACQUIRED])]
        [,TERM={ADR|FIRST|ID|NEXT|LOCAL|UNIQUE}]
        [,TRMADDR=([reg1][,reg2])]
        [,XLATEID=UNIQUE]
```

CTYPE=LOCATE
Requests the address of a terminal entry or a system entry in the TCT and optionally, either the address of an LDC entry in the system LDC table, or the unique compound name that identifies the object in the whole network.

DOMAIN={LOCAL|REMOTE|ALL|SYSTEM|SESSIONS}
Specifies the scope of the search to be carried out and implies the type of object to be found. The default is DOMAIN=LOCAL.

LOCAL
Locates a full TCTTE describing either a terminal belonging to this CICS region or a session connecting this region to another. The address of the TCTTE is returned in field TCATPTA. (Alternatively, a system entry (TCTSE) may be found, see TERM=ID on page 242.

REMOTE
Locates a model TCTTE describing a terminal belonging to another CICS region. The address of the TCTTE is returned in field TCATPTA. This TCTTE contains all the known attributes of the remote terminal and the field TCTTESKA in the model points to the skeleton terminal entry, a small control block identifying the remote terminal to the home region.

ALL
Locates either a full (belonging to the home region) or a model (describing a remotely owned terminal) TCTTE. The address of the full or model TCTTE is returned in field TCATPTA, and you must determine which type of TCTTE it is by testing its internal flags.

SYSTEM
Locates a TCTSE (system entry) that identifies one named route to a CICS region in the network, in this or another processor. There is always a local system entry that names the home region, and there may be indirect system entries that own terminals, but imply routing of all messages via an intermediate region. The address of the TCTSE is returned in field TCATPTA.

SESSIONS
Locates a TCTTE related to a session with an identified remote region. The address of the TCTTE is returned in field TCATPTA. Only the TERM=FIRST and TERM=NEXT formats are supported.

ERROR=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if an error occurs. Errors passed to this exit routine are those not handled by INVADDR, INVID, INVREQ, or INVLDC.

INVADDR=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the address specified in TCATPTA is not within the appropriate part of the terminal control table, not properly aligned, or zero for a DOMAIN=SESSIONS request. This operand is only applicable when an address is required in TCATPTA.

INVID=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the identifier specified cannot be located in the appropriate table. This operand is applicable with TERM=ID, TERM=UNIQUE, and NETNAME=ID.

LASTTRM=symbolic-address
Specifies the entry label of the user-written routine. Control is to be passed to it if the address that was preset in TCATPTA was:

- That of the last entry in the specified domain of the terminal control table with TERM=NEXT or NETNAME=NEXT; or,
- If the domain is empty, with TERM=FIRST or NETNAME=FIRST.

LDC={DEFAULT|YES}

Requests LDC information (the mnemonic, the numeric value, and/or the entry in the system LDC table or the extended local LDC list) associated with a specified TCTTE. If the LDC mnemonic is found, CICS returns (in TCATPLDA) the address of the LDC entry and (in TCATPLDC), the LDC numeric value. The LDC operand causes CICS to search the local LDC table for the LDC mnemonic. If the LDC mnemonic is found in the local table, the LDC numeric value is supplied from the local table. If the local table does not have the numeric value, the LDC value is taken from the system table. TCATPTA can be preloaded with the address of the TCTTE to be used; if TCATPTA is preloaded, the TERM operand must be given a value of ADR, or allowed to default.

Notes:

1. The LDC operand does not apply to 3614 logical units.
2. If an extended local LDC list exists for the terminal specified in the LDC operand, TCATPLDA is set to point to the extended local LDC list entry.

DEFAULT

Indicates that the default LDC is to be determined for the specified TCTTE. The default is the first LDC in the LDC list associated with the TCTTE. The default LDC mnemonic is returned in TCATPLDM, the numeric value in TCATPLDC, and the address of the LDC entry in the system LDC table or the extended local LDC list in TCATPLDA. If the default cannot be located, TCATPLDM is set to blanks, and TCATPLDC and TCATPLDA are set to binary zeros.

YES

Indicates that the two-character LDC mnemonic to be used has been preloaded in TCATPLDM. If TCATPLDM is set to blanks, the default LDC (as explained in DEFAULT below) is used; the mnemonic of the default is returned in TCATPLDM along with the other LDC information located. If the LDC cannot be located, TCATPLDC and TCATPLDA are set to binary zeros.

NETNAME={FIRST|NEXT|ID}

Specifies that a TCTTE is to be found in the local region, by reference to its node initialization block (NIB) description. The DOMAIN operand must have a value of LOCAL or be allowed to default, and if the TERM operand is specifies it will be ignored. The values that can be specified for NETNAME are:

FIRST

Returns the address of the TCTTE associated with the first NIB descriptor in field TCATPTA.

NEXT

Returns the address of the TCTTE associated with the next NIB descriptor in field TCATPTA. Note that, on invocation, the field TCATPTA must contain the address of a TCTTE that has an NIB.

ID On invocation, the field TCATPTA must contain the address of an eight-byte field containing the VTAM netname of a terminal or session. The address of the first TCTTE will be returned in field TCATPTA.

NORESP=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the required operation was performed successfully. The address of the located entry is returned in TCATPTA. NORESP signifies normal response.

SELECT=parameter list
Is used when attempting to locate the address of a terminal or session entry in the TCT if TERM=FIRST or TERM=NEXT is specified. The SELECT option is ignored if you specify any other TERM value, or if you use it when locating other types of entry, for example, a system entry. This keyword enables the user to specify a set of conditional states, at least one of which must be satisfied for the locate to be successful. The conditions must apply at the time the locate call is made, and CICS will carry out the tests necessary to determine whether the terminal or session entry reflects the state required. For example, if you code a "locate next" with a set of conditions specified by the SELECT keyword, you are asking CICS to find the next terminal or session which satisfies one of those conditions. This use of SELECT reduces substantially the path length of a search for a terminal or session of certain characteristics compared with a series of locates and individual tests in physical sequence.

In the list that follows, one or more of the options may be specified. Note that certain values, for example INSRV and OUTSRV, are mutually exclusive. If both are coded, CICS will ignore the value coded second.

Note that a similar facility is provided at command level by the EXEC CICS INQUIRE command, described in "Chapter 5.9. Examining and modifying resource attributes" on page 423.

INTLOG
Allows CICS to generate a request to create a session to the ACF/VTAM terminal.

NOINTLOG
Prevents CICS from creating a session to the ACF/VTAM terminal.

INSRV
A terminal or session that is in service. This means that CICS can receive input, including binds, from the terminal or session.

OUTSRV
Out of service prevents the terminal from either receiving messages (or binds) or transmitting input.

VTAM
Only terminals/sessions supporting VTAM are required.

PRIMARY
Sessions marked as primary normally receive before sending.

SECONDARY
Sessions marked as secondary normally send before receiving.

RELEASED
A session is not, or is no longer, bound.

ACQUIRED
The session is currently bound.

STATUS = parameter list
See SELECT keyword above for the values that you can specify for STATUS. The description of the SELECT keyword is also valid for STATUS, but when you use the STATUS keyword **all** of the conditions must be satisfied for a successful locate.

This operand is ignored in either of the following cases:

- It is coded when locating other types of entry; for example, a system entry.
- TERM = ID is coded (see the TERM operand of this macro).

TERM = {ADR|FIRST|ID|NEXT|LOCAL|UNIQUE}
Specifies the format of the data by which the entry is to be located. If the NETNAME operand is specified, the TERM operand will be ignored. The default is TERM = ADR.

ADR
Indicates that no searching will be done. The address in field TCATPTA on invocation will be validated according to the value specified for the DOMAIN operand. TERM = ADR must be specified or allowed to default if the LDC operand is specified. TERM = ADR is valid with a DOMAIN value of LOCAL, REMOTE, or ALL.

FIRST
Finds the first entry according to the value specified for the DOMAIN operand as follows:

- For DOMAIN = LOCAL, REMOTE, or ALL, it finds the first terminal or session entry in the specified domain.
- For DOMAIN = SYSTEM, it finds the first system entry.
- For DOMAIN = SESSIONS, it finds the first session in the TCTSE whose address is specified in field TCATPTA on invocation.

NEXT
Given an entry address in field TCATPTA, finds the next entry of the same type in that domain. If the domain is local, either all system entries, or all the local region TCTTEs will be scanned according to the type of the entry provided. If DOMAIN has a value of ALL, the local region TCTTEs will be scanned before any model (remote) TCTTEs. If DOMAIN has a value of SESSIONS, only sessions under the same system entry as that supplied in TCATPTA are returned.

If field TCATPTA is set to zero, the effect will be the same as specifying TERM = FIRST.

ID Finds the entry containing the four-byte identifier of a terminal, session, or system that has been specified in field TCATPTA. The answer is returned in field TCATPTA. Only DOMAIN values of LOCAL, REMOTE, ALL, or SYSTEM are valid with this operand.

LOCAL
Finds the local system entry (TCTSE) that names the home region. The only DOMAIN value that may be specified with TERM=LOCAL is SYSTEM.

UNIQUE
Finds the full or model TCTTE corresponding to the two-part identifier specified. On invocation, field TCATPAPL must contain the eight-byte netname of a CICS region, and field TCATPTA must contain the terminal identifier of the terminal in that region. This two-part identification uniquely identifies any terminal in the network of CICS regions. The address of the TCTTE will be returned in field TCATPTA. Only DOMAIN values of REMOTE or ALL are valid with this operand.

TRMADDR=([reg1][,reg2])
Use of the keyword TRMADDR enables the caller to avoid both saving the current TCTTE address into, and restoring the located address from, TCATPTA.

Registers 1 and 2 are general purpose registers.

1. "Reg1" contains the entry name or holds the address of the current entry.

   If "reg1" is omitted, you must save the details into TCATPTA yourself.

2. "Reg2" will be set on exit to hold the address of the located entry, or zero.

   If "reg2" is omitted, its value will default to the value specified as "reg1". If "reg1" is also omitted, you must obtain the located entry address from TCATPTA yourself.

XLATEID=UNIQUE
On successful location of a TCTTE, the unique identification of the terminal is returned as well as the address of the TCTTE. Field TCATPTA will contain the netname of the CICS region, and field TCATPRMT will contain the identifier of the terminal in its own region. If the locate operation finds a system entry, no unique identification is returned.

Changing status — DFHTC CTYPE=STATUS

The DFHTC CTYPE=STATUS macro instruction should be used to perform any change of status, instead of directly altering bits in the TCTTE. You should be aware that, when CICS emergency restart procedures are invoked following a catastrophic system failure, the status of each logical unit is set to the specification given in the original terminal control table. This is because none of the dynamic changes are retained after the failure.

Don't use DFHTC CTYPE=STATUS against a surrogate TCTTE. This may cause status logic errors to occur.

```
DFHTC   CTYPE=STATUS
        [,ERROR=symbolic-address]
                 [,INVADDR=symbolic-address]
                 [,INVID=symbolic-address]
                 [,INVLDC=symbolic-address]
                 [,INVREQ=symbolic-address]
                 [,LASTTRM=symbolic-address]
                 [,NORESP=symbolic-address]
                 [,LDC=YES]
                 [,STATUS=([INSRV|OUTSRV]
                           [,TRANSCEIVE|TRANSACTION|RECEIVE|
                             INPUT|NOPOLL]
                  [,PAGE|AUTOPAGE]
                  [,ACQUIRE|RELEASE]
                  [,COLD])]
        [,TERM={FIRST|NEXT|ID}]
```

CTYPE=STATUS
Specifies that the status of a logical unit or an LDC is to be changed and/or the terminal entry is to be located.

ERROR, INVADDR, INVID, INVLDC, INVREQ, LASTTRM, and NORESP=symbolic-address
Are used to test the CICS response to the request for STATUS. These operands can be specified in this macro instruction or in a DFHTC CTYPE=CHECK macro instruction. These operands are defined in the description of the DFHTC CTYPE=CHECK macro instruction. See "Test CICS response to CTYPE requests — DFHTC CTYPE=CHECK" on page 246.

LDC=YES
Requests the status change of an LDC represented by the specified LDC mnemonic in the system LDC table, or in the extended local LDC list. You should specify LDC=YES and TERM= to change the status of an entry in the extended local LDC list; otherwise the system LDC list will be searched. You specify the LDC mnemonic in TCATPLDV before issuing this request.

The LDC operand can only be specified with PAGE/AUTOPAGE status change requests. This operand does not apply to 3614 logical units.

Note: If you specify LDC=YES and TERM=, the INVLDC condition will be raised if the extended local LDC list does not exist, or if the LDC specified does not exist in that list. The system LDC table is not searched if you specify TERM=.

STATUS=logical-unit-status
Requests that the status of a logical unit or an LDC be changed.

The following parameters indicate the status changes for the specified logical units or the LDC: INSRV, OUTSRV, TRANSCEIVE, TRANSACTION, RECEIVE, INPUT, NOPOLL, PAGE, AUTOPAGE, ACQUIRE, RELEASE, COLD.

The meanings of these status changes are as follows:

An **INSRV** (in-service) logical unit is one that can transmit and/or receive data with CICS.

An **OUTSRV** (out of service) logical unit is one that can neither transmit to nor receive data from CICS.

A logical unit in **TRANSCEIVE** status is a TRANSACTION terminal to which messages are sent automatically by the user. The automatic transaction initiation, which is created by a transient data destination reaching a trigger level or by a time interval (such as message switching), sets a condition in an appropriate terminal control table terminal entry (TCTTE). If the terminal status is TRANSCEIVE and if there is no transaction at the terminal, terminal control initiates the user-defined task. The function of this task is to send messages to the terminal.

A logical unit in **TRANSACTION** status is used in the processing of transactions such as inquiries or order entries, but cannot receive automatic output.

A logical unit in **RECEIVE** status is one to which messages can be sent, but from which no input is allowed.

A logical unit in **INPUT** status is one which can send messages to CICS but cannot receive messages from CICS. Note that system messages may be routed to an input logical unit under certain conditions, for example, when an invalid transaction ID has been entered. This causes DFHZNAC to be scheduled. You should code a node error program to perform any user-required action.

NOPOLL indicates that CICS is no longer to attempt to read from the logical unit.

PAGE indicates that all requests to process data from the paging supervisor are to be paged, unless specified otherwise in the DFHBMS macro or command. When paging, the first page from the paging supervisor is written when the logical unit becomes available. All subsequent pages in a page series are written on request of the logical unit (from the operator, if so designed) using paging commands.

AUTOPAGE indicates that all requests to process data from the page supervisor are to be automatically paged unless specified otherwise in the DFHBMS macro or command. When autopaging, the page supervisor writes all pages in a page series automatically. Requests to write data directly to the logical unit are not controlled by the PAGE or AUTOPAGE parameters, because the page supervisor is not used for direct output.

Note: PAGE and AUTOPAGE only apply to LDC=YES or to TERM=.

ACQUIRE indicates that the specified logical unit is to be acquired from ACF/VTAM.

RELEASE indicates that the specified logical unit is to be released to ACF/VTAM.

ACQUIRE,COLD indicates that the specified logical unit is to be acquired from ACF/VTAM but that message resynchronization is not to be attempted with the logical unit. This specification is enforced in the case of a 3270-system terminal, the interactive logical unit (3767, 3770), and the batch logical unit (3770).

TERM={FIRST|NEXT|ID}
Indicates that a terminal entry is to be located and its status changed. If LDC=YES is specified with TERM=, the extended local LDC list for that terminal (if located) is changed, not the terminal entry. The address is returned in the TCATPTA. If both the TERM and LDC operands are omitted, TCATPTA is assumed to contain the address of the terminal entry for which the STATUS request is being made.

FIRST
Indicates that the first terminal entry in the terminal control table is to be located.

NEXT
Indicates that the terminal entry following that specified in TCATPTA is to be located. If TCATPTA is preset with binary zeros, the first terminal entry is located.

ID Indicates that the terminal entry with a specified terminal ID is to be located. TCATPTA must be preset with the terminal ID (left-justified) and padded with blanks (X'40') to fill the four-character field.

If this operand is omitted, it is assumed that TCATPTA has been preset with the address of the terminal entry to be changed.

Test CICS response to CTYPE requests — DFHTC CTYPE=CHECK

The general format of the DFHTC macro instruction to test the CICS response to a preceding DFHTC request for LOCATE or STATUS is:

```
DFHTC   CTYPE=CHECK
        [,ERROR=symbolic-address]
        [,INVADDR=symbolic-address]
        [,INVID=symbolic-address]
        [,INVLDC=symbolic-address]
        [,INVREQ=symbolic-address]
        [,LASTTRM=symbolic-address]
        [,NORESP=symbolic-address]
```

CTYPE=CHECK
Indicates that the CICS response to a DFHTC CTYPE=LOCATE or DFHTC CTYPE=STATUS request is to be checked.

ERROR=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if an error occurs. Errors passed to this exit routine are those not handled by INVADDR, INVID, INVREQ, or INVLDC.

INVADDR=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the address specified in TCATPTA is not within the limits of the terminal control table, properly aligned on a fullword boundary, or zero for a TERM=NEXT form. This operand is only applicable when an address is required in TCATPTA.

INVID=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the terminal ID specified in TCATPTA is not located in the TCT. This operand is only applicable to TERM=ID.

INVLDC=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the LDC mnemonic is not found in the system LDC table or in the extended local LDC list. This operand is only applicable to paging status requests for LDCs.

INVREQ=symbolic-address
Specifies the entry label of the user-written routine to which control is passed if an erroneous bit setting is deleted during execution of the macro instruction.

LASTTRM=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the address that was preset in TCATPTA was that of the last terminal entry in the table. This operand is only applicable to TERM=NEXT.

NORESP=symbolic-address
Specifies the entry label of the user-written routine to which control is to be passed if the required operation was performed successfully. NORESP signifies normal response.

Command option for logical units — DFHTC CTYPE=COMMAND

You can use the DFHTC CTYPE=COMMAND macro instruction to transmit indicators from CICS to the logical unit application program. You should use the indicator interface to request an ACF/VTAM function, rather than directly altering bits in the TCTTE, which could lead to unpredictable results if any changes are made in the TCTTE internal structure in the future.

```
DFHTC   CTYPE=(COMMAND [,WAIT])
        [,COMMAND=indicator]
```

Note: This macro instruction is not valid for ACF/VTAM-supported 3270s, and will cause an abend if so used. The macro may, however, be used for logical units in 3270-compatibility mode.

CTYPE=(COMMAND[,WAIT])
Specifies that an ACF/VTAM indicator is to be transmitted. The indicator is specified in the COMMAND operand.

COMMAND=indicator
Specifies the type of indicator to be sent. The following SNA data flow control and session control commands can be specified:

BID
Requests permission to start a bracket for a particular TCTTE. CICS uses the BID command as part of the ATI process for all logical units that use bracket protocol.

CANCEL
Requests the receiver to ignore the chained message it is currently receiving.

CHASE
Forces any pending responses to be returned to CICS.

CLEAR
Resets all sequence numbers to zero, and puts the connection in the data flow reset state. No data may be sent to, or received from, the logical unit until the series definition table (SDT) command has been sent. Only session control commands (STSN and SDT) may be sent when the connection is in data traffic reset state.

QC
Quiesce-complete is used by a node to respond to a QEC request to indicate that it is now in quiesce state.

QEC
Quiesce-at-end-of-chain requests that a logical unit that is either out of service or in receive-only mode be quiesced (but not released from CICS control) after the message currently being transmitted from it is received.

RQ
Release-quiesce is used by the node that issued the QEC request, and removes that node from the quiesce state.

SDT
Start-data-traffic removes the specified connection from the data flow reset state so that the data and data-flow indicators may be sent.

SHUTD
Shutdown indicates that an end-of-day condition has been reached. CICS sends this command during termination.

SIGNAL
Causes an expedited signal to be sent to the terminal.

STSN
Set-and-test sequence number is used during recovery from a failure to determine whether any in-flight messages were lost.

CICS always sends ACF/VTAM indicators with definite FME/DR1 response protocol requested. DFHZCP calls the appropriate routine and returns control to the requester when the response is received.

Chapter 4.8. The user program for automatic installation of terminals

You can use the CEDA DEFINE command to define VTAM-connected resources to your CICS system. This transaction puts your definitions onto the CICS system definition file which sets up entries in the terminal control table (TCT). This method is described in the *CICS/MVS Resource Definition (Online)* manual. As an alternative, you can allow CICS to create entries in the TCT whenever VTAM resources request connection to CICS. The automatic installation (autoinstall) user program can control this process. A particular advantage of autoinstall is that the resource occupies storage in the TCT only while it is connected to CICS. This chapter tells you how to use the CICS-supplied autoinstall program and how to extend it to suit your own purposes. Before you read this chapter, you should have read the sections in the *CICS/MVS Resource Definition (Online)* manual that describe the CEDA commands that create the environment in which your user program can work.

If you choose automatic installation for some or all of your terminals, you must:

- Define your terminals in VTAM so that they match the autoinstall model definitions in CICS.
- Use the default sample program or write your own user program, using the sample in this chapter as a basis if you want to. (You can write a new program if the sample does not meet your needs, but it might be a good idea to try a sample-based program first.)
- Enable the CICS AUTOINSTALL function by using DFHSIT or CEMT.
- Enable your user program using DFHSIT or CEMT.
- Define some AUTOINSTALL models.

Start simply with autoinstall

The autoinstall user program is a command level program. You could execute the sample program with CEDF by writing a simple program to build a COMMAREA and EXEC CICS LINK to one of the sample autoinstall programs. You could then change the parameters dynamically to familiarize yourself with the program execution and the contents of the various DSECT data fields before writing your version. After you have determined which devices are to be autoinstalled, defined the models (TYPETERM and TERMINAL), and installed them in the CSD, be sure to remove any DFHTCT macro definitions for these devices.

A useful starting technique

The following technique has proved useful in dealing with the complexities of a network with multiple types of terminal device. The box contains background information needed to understand the technique.

VTAM bind requests

The VTAM LOGMODE table consists of entries called MODEENTs. Each MODEENT defines a set of characteristics that can be used to bind a session. With each bind request (CINIT) that CICS receives, VTAM sends MODEENT data in a request unit (RU). The RU is pointed to by fullword 5 in the parameter list passed to the user autoinstall program (see page 255) and is described in the *IBM SNA Reference Summary*, GA27-3136. The RU data includes the SNA logical unit (LU) type of the requester.

On receipt of the bind request, the CICS transaction CQRY (invoking module DFHQRY) queries the requested terminal and determines the terminal attributes.

Given that the CINIT RU contains the LU type and that the response to CQRY can provide many of the terminal attributes, within your autoinstall user program, do the following:

1. Look at the 13th byte of the BIND image in the CINIT RU. This will contain the LU type (BIND image begins 12 bytes into CINIT RU).
2. Choose a model TYPETERM according to the LU type found in the RU. For each LU type, there is a CICS-supplied model TYPETERM defined with QUERY(ALL), which means that all attributes are acquired from the terminal by CQRY. The model TYPETERMs are listed in Appendix B of the *CICS/MVS Resource Definition (Online)* manual. For example, if the device is an LU2 device, set the model name field to DFHLU2 (where QUERY(YES) is defined in the DFHLU2 TYPETERM) and the query structured field takes care of the rest.

For devices that do not support query by the CQRY transaction, the program must make a decision based on the terminal types in the installation. For example, it might be decided to support these as model 2 devices only.

Implementation checklist

You may need to get a VTAM specialist to help you plan for, and implement, autoinstall.

If you are installing CICS for the first time, you should read the *CICS/MVS Resource Definition (Online)* manual for help in getting RDO started and the first terminal autoinstalled.

If you have an established CICS system:

1. Decide whether to implement autoinstall. Read this chapter, and the information in the *CICS/MVS Resource Definition (Online)* manual. Evaluate the benefits and the costs for your system.
2. Decide which devices to autoinstall. Read this chapter and analyze the requirements of your network. Consider the current use of ATI, AUTOCONNECT, TCTUAs, TLTs, MRO and ISC.

3. If you have not yet started to use RDO for terminals:

   - Read about RDO in the *CICS/MVS Resource Definition (Online)* manual, and plan your migration using the chapter on "Migrating the TCT to the CSD".
   - Migrate your DFHTCT macros to the CSD, and study the resulting definitions.

4. Assess the suitability of the definitions listed in the "IBM-Supplied Resource Definitions, Groups, Lists" appendix in the *CICS/MVS Resource Definition (Online)* manual.
5. Create the autoinstall models and TYPETERMs that you will need, trying to minimize the number, so that the autoinstall program can be as simple as possible.
6. Read about the QUERY function in the *CICS/MVS Resource Definition (Online)* manual.
7. Ensure that your VTAM LOGMODE table entries are correct.
8. Consider your TERMINAL naming conventions, and decide whether constant, predictable names are important for your terminals.
9. Design and write an autoinstall program.
10. Test that it will autoinstall terminals you want it to allow, and that it will not autoinstall any terminals you want it to prevent from logging on to the CICS system. (Unless explicitly prevented by the program, any terminal that has no TCT entry is allowed.)
11. If CICS can find no models compatible with the VTAM information describing the resource, you can create a test autoinstall program that forces the model name (AUTINSTNAME) you want. With a VTAM buffer trace running, try to log the device on to CICS. If CICS does not attempt to send a BIND, check the following:

    - Does the model TERMINAL refer to the correct TYPETERM? (Or, alternatively, is the TYPETERM in question referred to by the correct TERMINAL definition?)
    - Is the TERMINAL definition AUTINSTMODEL(YES or ONLY)?
    - Have you installed the groups containing the autoinstall models (TERMINAL and TYPETERM definitions)?

    If CICS attempts to BIND, compare the device's CINIT RU to the CICS BIND, and make corrections accordingly.

    It is very important that you ensure that the VTAM LOGMODE table entries for your terminals are correct, rather than define new autoinstall models to fit incorrectly coded entries. Bear in mind, while you are testing, that CICS autoinstall will not work if a LOGMODE entry is incorrectly coded.

    Note that you cannot *force* device attributes by specifying them in the TYPETERM definition. For autoinstall, the attributes defined in the LOGMODE entry must *match* those defined in the model; otherwise the model will not be selected. You cannot define a terminal one way to VTAM and another way to CICS.

12. If it *appears* that TCT entries are being autoinstalled successfully, ensure that you are not actually installing them by any other method:
    - Check that the groups included in the list named in the GRPLIST operand of DFHSIT do not include explicit TERMINAL definitions for the terminals
    - Check that MIGRATE=COMPLETE is coded in the DFHTCT TYPE=INITIAL macro.

Using the query facility with autoinstall

The QUERY attribute of the CEDA DEFINE TYPETERM transaction allows you to leave some features of your terminals undefined until they are connected, when information about these features can be obtained from the device.

The transaction name is CQRY (module name DFHQRY). It is invoked prior to the "good morning" message to issue the SNA query structured field command. This acquires the attributes of the terminal device that has just been connected to CICS.

When QUERY(ALL|COLD) is used in the CEDA DEFINE TYPETERM transaction, the query process resets all of the flags that can be set by specific definitions in the the CEDA DEFINE TERMINAL transaction. These include:

- Extended data stream support
- Color
- PSS (program symbol sets)
- Highlight
- Field validation
- Partitions
- MSR control field
- The following Kanji functions:
  - Outlining
  - Mixed fields
  - Background transparency.

If the query fails, these fields will remain set, and the screen size page size, ALTSCREEN size, and ALTPAGE size will remain unchanged. If the query completes without error, these fields will be updated.

However, you can use the CEDA DEFINE TERMINAL transaction to specify attributes supported by the device, and code QUERY(NO) in the associated TYPETERM.

Within a CICS trace, there are special trace entries produced to show how these fields were set after the query. See the *CICS/MVS Problem Determination Guide* for a detailed layout of these special trace entries. They are FA (BMS) traces with a REQD of 81 or 82.

Additional information on this capability may be found in the "Communication Resources" chapter in the *CICS/MVS Resource Definition (Online)* manual.

Only devices that are able to respond to the Write Structured Field Query, an extended data stream command, may be defined with the QUERY attribute. In addition, when using facilities such as protocol converters or the older versions of VM/CMS PASSTHRU, the use of extended data stream commands may not be supported.

Coding entries in the VTAM LOGON mode table

Appendix D, "Coding entries in the VTAM LOGON mode table" on page 535 shows you what you must code in your VTAM logmode table for a terminal that you want to be able to install automatically.

CICS uses the logmode data in the VTAM logmode table when processing an autoinstall request. For successful automatic installation, this information must be correct because CICS conforms to VTAM standards.

The tables in Appendix D, "Coding entries in the VTAM LOGON mode table" on page 535 show what you must code in the VTAM MODEENT macros if you want to use autoinstall. Between them they show the values that must be specified for each of the operands of the MODEENT macro.

Some of the examples in the appendix correspond exactly to entries in the IBM-supplied logon mode table called ISTINCLM. Where this is so, the table gives the name of the entry in ISTINCLM.

The figures showing PSERVIC settings indicate fields called aaaaaaaa, bbbbbbbb, and so on. The contents of these vary according to certain specifications of attributes of LUTYPE2 and LUTYPE3 terminals.

The user program

As well as managing your resource definition, your user program can perform any other processes you want to carry out at this time. Its access to the command-level interface is that of a normal, nonterminal user task. Some possible uses are listed on page 260.

The user program receives control when:

- An autoinstall INSTALL request is being processed
- An autoinstall INSTALL has failed
- An autoinstall DELETE request is being processed.

On each invocation of the user program, a parameter list is passed (using the COMMAREA mechanism of the EXEC CICS LINK command), describing the function being performed (INSTALL or DELETE), and providing data relevant to the particular event.

The two events are now described in detail.

The user program at INSTALL

You enable and disable the autoinstall option, and change the name of the user program if you want to, by using:

- The AUTINST operand of DFHSIT

  OR

- The CEMT INQUIRE | SET AUTOINSTALL command.

(See the *CICS/MVS Resource Definition (Macro)* manual for information about DFHSIT and the *CICS/MVS CICS-Supplied Transactions* manual for information about CEMT.)

If autoinstall is operative, the user program receives control at INSTALL when:

1. A VTAM logon request has been received from a resource eligible for automatic installation whose NETNAME is not in the TCT,

   AND

2. Autoinstall processing has completed to a point where information (a terminal identifier and autoinstall model name) from the user program is required in order to proceed.

In this event, the user program is passed a pointer (through DFHEICAP) to a parameter list that consists of five contiguous fullwords. Figure 21 on page 255 illustrates these fields.

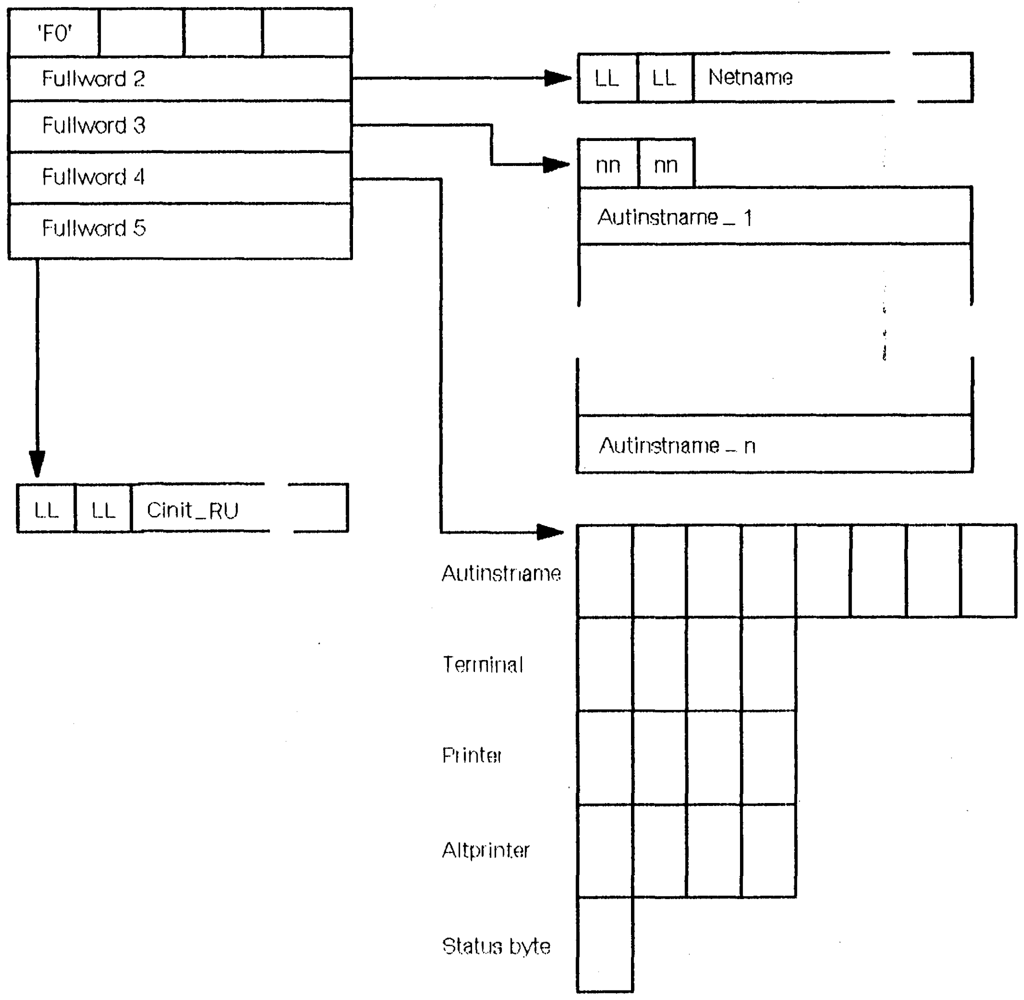


Figure 21. User program at INSTALL parameter list

The fullwords in Figure 21 have the following meanings: Fullword number:

1. Function field. Byte 1 indicates the request type. (This is character '0' for INSTALL.) The remaining three bytes are reserved.
2. Pointer to identifier field. The identifier field consists of a two-byte length field, followed by the NETNAME of the resource requesting LOGON.
3. Pointer to an array of names of eligible autoinstall models. The array is preceded by a two-byte field describing the number of eight-byte name elements in the array. If there are no elements in the array, the number field is set to zero. This list of names is further described under "Processing."
4. Pointer to return information field. This 21-byte field is set to blanks on input. It is storage where you may place information for return to the calling program. The setting of values in this field is discussed later.
5. Pointer to VTAM LOGON data (the CINIT request unit). The data is preceded by a two-byte length field, indicating the length of the CINIT request unit, and includes the three-character NS header. The format of the CINIT request unit is described in the *IBM SNA Reference Summary* (GA27-3136).

Processing

At the INSTALL event, the user program is responsible for allowing or denying the connection of a new terminal resource to the CICS system. This decision may be based on a number of installation-dependent factors, such as security, or the total number of connected terminals. CICS takes no part in any such checking. You decide whether any such checking takes place, and how it is done. The user program provides you with the means of implementing your design.

If the user program decides that the autoinstall request is to proceed, then the values it must return to CICS are:

- The name of an autoinstall model (AUTINSTNAME)
- An associated terminal identifier (TERMINAL).

These values are both used as input to the INSTALL request. In addition, the user program must indicate in the status byte if the event is to proceed.

How the model name is selected

For a device logging on, the model name to be used is normally selected from the array that is passed in the parameter list (as described in point 3 above). The autoinstall models in the array are selected by CICS from the complete list of terminal models because of their compatibility with the VTAM information describing the resource. The complete list of autoinstall models available to CICS at any time will comprise all the definitions with AUTINSTMODEL(YES) and AUTINSTMODEL(ONLY) that have been installed, both by the GRPLIST on CICS cold start, and by INSTALL GROUP commands issued by CEDA. The *CICS/MVS Resource Definition (Online)* manual describes the definition of models.

Figure 45 on page 535 gives you the information to work out which model types could be included in a subset of models passed to the user program when a particular terminal attempts to install. The subset is determined by the VTAM characteristics of the device attempting to log on. The number in the right-hand column of the figure indicates the selection of the subset from the full list. When a terminal with a given combination of DEVICE, SESSIONTYPE and TERMMODEL attempts to log on, the subset of matching models passed to the user program includes all the models with DEVICE, SESSIONTYPE and TERMMODEL values that have a corresponding VTAM category number in the right-hand column of the table.

For example, if a 3270 printer attempts to autoinstall, the subset of matching models includes all the types in VTAM category 2 that you have defined as models. This subset could include any of the following:

```
DEVICE(3270) TERMMODEL(2)
DEVICE(3270) TERMMODEL(1)
DEVICE(3270P) TERMMODEL(2)
DEVICE(3270P) TERMMODEL(1)
DEVICE(3275) TERMMODEL(2)
DEVICE(3275) TERMMODEL(1)
```

Then the user program selects one model that is suitable for the device logging on. This model is used to build the TCT table entry, and determines the CICS attributes of the automatically installed terminal. (The IBM-supplied version of DFHZATDX is coded to select the first one in the subset.)

If CICS can find no models compatible with the VTAM information describing the resource, word 3 of the parameter list points to an empty array (the initial half-byte is set to zero). If this occurs, the error message DFH5987I contains a 'BEST FAILURE' model name as a diagnostic aid.

If the user program returns a model name that is not in the subset passed to the user program, CICS cannot guarantee what will happen when further processing takes place. It is the user's responsibility to determine the effect of associating any particular logon request with a particular model name, no interface being provided to the in-storage "model" objects.

| If CICS finds no suitable subset of models, an empty list, pointed at by fullword 3 of the parameter list (see Figure 21 on page 255), is passed to the user program. If the user program returns a nonzero status value in these circumstances, the resulting error message DFH5987I will still contain a 'best failure' model name, but this is provided for diagnostic purposes only. More information about the DFH5987I error message is provided on page 258.

In addition to the required information, the user may also supply PRINTER, and ALTPRINTER values, to be used as additional information on the INSTALL request.

If the program decides that the autoinstall request is to be rejected, it should not | return any information in the return information fields pointed to by fullword 4 of the parameter list (see Figure 21 on page 255).

Returning information

The format of the information returned in the fields listed below is defined in the *CICS/MVS Resource Definition (Online)* manual.

The fields are:

- AUTINSTNAME — 8 bytes
- TERMINAL — 4 bytes
- PRINTER — 4 bytes
- ALTPRINTER — 4 bytes
- status byte — 1 byte.

To return information, the user program sets the required values into the return information field (format described above). The status field must be set to binary zeros.

Having completed processing, the user program must return to CICS by issuing an EXEC CICS RETURN command.

CICS action on return

When CICS receives control back from the user program, it examines the return information status field. If this is zero, and if the other required information supplied is satisfactory, CICS will schedule the new resource for OPNDST in order to complete the logon request. If VTAM refuses to set up the session, the user program will be driven again, as though a DELETE had occurred. (See "The user program at DELETE" on page 259 for details.) This is to allow the user to free any allocations made on the assumption that this INSTALL request would succeed — terminal identifiers, for example.

If the return information status field is not zero (or if it was, and the request failed for some reason), CICS rejects the connection request in the same way as it rejects an attempt by an unknown terminal to LOGON to CICS when autoinstall is not enabled.

For all autoinstall activity, messages are written to the transient data destination CADL. If an INSTALL fails, a message is sent to CADL, with a reason code. You can therefore check the output from CADL to find out why an autoinstall request failed.

Details of the best failing match between a model and the BIND-image are written to the CADL transient data destination if an autoinstall attempt fails for lack of an exact match.

The message takes the following form:

```
DFH5987I BEST FAILURE FOR NETNAME: nnnnnnnn,
         WAS MODEL_NAME: mmmmmmmm,
         MISMATCH BITS: xxxxxxxx...
```

where

- 'nnnnnnnn' is the netname of the LU which failed to logon.
- 'mmmmmmmm' is the name of model that gave the best failure (that is, the one that had the fewest bits different from the BIND-image supplied by VTAM).
- 'xxxxxxxx...' is a string of hexadecimal digits, where 'xx' represents one byte, and each byte position represents the corresponding byte position in the BIND-image. A bit set to '1' indicates a mismatch in that position between the BIND-image from VTAM and the BIND-image associated with the model.

A suggested course of action for the system programmer responsible is as follows:

1. Determine whether a model such as 'mmmmmmmm' is suitable. If there are several models which have identical BIND-images, differing only in end-user options, such as OPERSEC, then only the first such model is named in the above message. It will be up to the user-program to make the choice, when the logmode table entry is corrected.
2. Identify the entry in the VTAM logmode tables that is being used.
3. Check that this logmode table entry is not successfully in use with other applications, so that to change it might cause this other use of it to fail.

4. Amend the logmode table entry by switching the bits corresponding to '1' bits in the mismatch string. That is, if the bit in the VTAM bind image corresponding to the bit position set to '1' in 'xxxxxxxx...' above is '1', set it to '0'; if it is '0', set it to '1'.

More information about the meaning of the various bits in a BIND-image, and some more references, may be found in *ACF/VTAM Programming*, SC27-0611, Appendix N. Details of the preparation of VTAM logmode table entries are given in *ACF/VTAM Customization*, SC27-0613.

The user program at DELETE

In order to provide symmetry of user control over the autoinstall process, the user program also receives control at DELETE. That is, when:

- A session with a previously automatically installed resource ends;

  OR

- An autoinstall request is accepted by the user program, but the subsequent INSTALL process fails for some reason.

To make it easier for you to write the user program, these two events may be considered identical. (There is no difference in the environment that exists, or in the actions that may need to be performed.)

Invoking the user program at DELETE enables the user to reverse the processes carried out at INSTALL. (For example, if the user program at INSTALL increments a count of the total number of automatically installed resources, the user program at DELETE should decrement that count.)

The user program is passed a pointer, by DFHEICAP, to a parameter list that consists of two contiguous fullwords. The fullwords have the following meanings:

1. Function field. Byte 1 indicates the request type (this is character '1' for DELETE). The remaining three bytes are reserved.
2. The terminal identifier of the deleted resource.

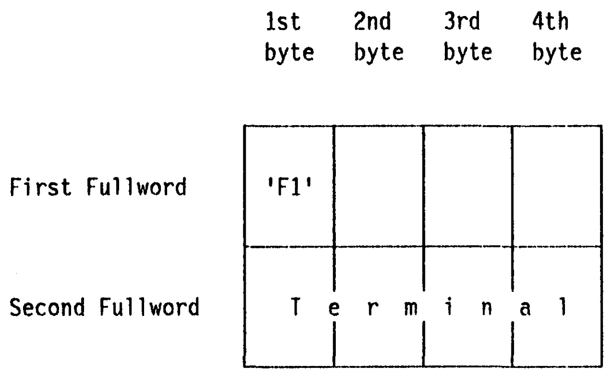


Figure 22. User program at DELETE parameter list

Note that the named resource has been deleted by the time the user program is invoked, and will not therefore be found by any TC LOCATE type functions.

Processing a rejection for a duplicate TERMID

The autoinstall user program is invoked as for a DELETE. To distinguish this invocation from a legitimate delete you can issue an EXEC CICS INQUIRE TERMINAL command. If a terminal entry still exists for the TERMID provided, the invocation represents a duplicate TERMID error. If the terminal entry no longer exists, the invocation is for a legitimate DELETE.

Testing and debugging your user program

To help you test the operation of your user program, you can run the program as a normal terminal-related application. Define your program and initiate it from a terminal. The parameter list passed to the program is described in "The user program at INSTALL" on page 254. You can construct a dummy parameter list in your test program, upon which operations can be performed. Running your program on a terminal before you use it properly means that you can use the EDF transaction to help debug your program. You may also make the program interactive, sending and receiving data from the terminal.

Example program

The example user program (name DFHZATDX), coded in assembler, is listed below. Programs that perform the same functions are then listed in COBOL and PL/I, followed by examples of how you could customize the user program. These sample programs are provided in source form in the CICS212.SAMPLIB library.

The example user program (assembler version) is supplied in the CICS212.SOURCE library. The module generated from this source is part of the pregenerated library shipped in CICS212.LOADLIB. This may be used unmodified, or it may be used as a base, to which user-dependent processing may be added. If you choose to alter the code in the example module, take a copy of the example and modify it. After modification, use DFHEITAL to translate, assemble and link-edit your module. Then put the load module into a user library that is concatenated before CICS212.LOADLIB in the DFHRPL statement. (This procedure applies to completely new modules as well as modified example modules.) For more information about this procedure, see the *CICS/MVS Operations Guide*. Do not overwrite the example with your customized module, because subsequent service may overwrite your module.

The default action of the example program, on INSTALL, is to select the first model in the list, and derive the terminal identifier from the last four nonblank characters of the NETNAME, set the status byte, and return to CICS. If there are no models in the list, it returns to CICS with no action.

The default action, on DELETE, is to address the passed parameter list, and return to CICS with no action.

You can customize the user program to carry out any processing that suits your installation. Examples of customization are given on page 272, after the example programs. Here are some customization suggestions.

Your user program could:

1. Count and limit the total number of logged-on terminals.
2. Count and limit the number of automatically installed terminals.
3. Keep utilization information about the terminal.
4. Map TERMINAL name and NETNAME.
5. Carry out general logging.
6. Handle special cases (for example, always allow certain terminals or users to logon).
7. Send messages to the operator.
8. Exercise network-wide control over autoinstall. A network-wide, global autoinstall user program may reside on one CICS system. When an autoinstall request is received by a user program on a remote CICS system, this global user program may be involved and data may be transferred from one user program to another.

Assembler example program

```
***********************************************************************
*                                                                     *
* MODULE NAME = DFHZATDX                                              *
*                                                                     *
* DESCRIPTIVE NAME = C.I.C.S. (VTAM) Autoinstall User Program         *
*                                                                     *
*                                                                     *
*                                                                     *
* FUNCTION = Provide user input to Autoinstall processing             *
*                                                                     *
*            This module is a component of ZCP.                       *
*                                                                     *
*            It is the default Autoinstall user program(i.e. if the   *
*            user does not wish to provide his own). It may be        *
*            used as a framework for the user to include his own      *
*            processing requirements.                                 *
*                                                                     *
*            It is called via an EXEC CICS LINK command, from         *
*            DFHZATD(the Autoinstall program).                        *
*                                                                     *
*            Input to the module is a parameter list addressed by     *
*            DFHEICAP.                                                *
*                                                                     *
*            The program is invoked when:                             *
*            1) An autoinstall INSTALL is in progress                 *
*            2) An autoinstall DELETE has just completed              *
*                                                                     *
*            The function to be performed is indicated via the passed *
*            parameter list.  This is evaluated during common initial- *
*            ization processing, and control passed to the appropriate*
*            routine.                                                 *
*                                                                     *
* Function 1 - autoinstall INSTALL                                    *
* --------------------------------                                    *
*                                                                     *
*    Parameter list:                                                  *
*                                                                     *
*    Fullword 1 =   Function field                                    *
*      Byte 1     -   Request type(X'F0' for INSTALL)                 *
*      Byte 2     -   Reserved                                        *
*      Byte 3     -   Reserved                                        *
*      Byte 4     -   Reserved                                        *
*    Fullword 2 ==> Netname                                           *
*    Fullword 3 ==> Model_name list                                   *
*    Fullword 4 ==> Return field                                      *
*    Fullword 5 ==> CINIT_RU                                          *
*                                                                     *
*    The format of the data is as follows:                            *
*                                                                     *
*    Netname - |x|Netname|                                            *
*    (where 'x' is 2 byte length of Netname)                          *
*                                                                     *
*    Model_name list - |y|Model1|Model2|...|Modeln|                   *
*    (where 'y' is a 2 byte number of names in the list; each name    *
*     element is 8 bytes in length; y may be zero, in which case no   *
*     list follows).                                                  *
*                                                                     *
*    Return field - |Model_name|TRMIDNT|PRINTTO|ALTPRT |Status|       *
*                   |8 bytes   |4 bytes|4 bytes|4 bytes|1 byte|       *
*    (A 21 byte field to return selected items and status information)*
*                                                                     *
*    CINIT_RU - |z|CINIT_RU|                                          *
*    (where 'z' is 2 byte length of CINIT_RU)                         *
*                                                                     *
*    The purpose of this function is to select a model name, and      *
*    corresponding terminal id to be used as input for an autoinstall *
*    resource 'builder' request.                                      *
*    Optionally, Printer and Altprinter values may be supplied.       *
*                                                                     *
*    The default action of this program on this event is a follows:   *
*    Model_name = the first name in the supplied list                 *
*    Terminal= the last 4 non-blank characters of the supplied NETNAME*
*    Printer = not supplied                                           *
*    Altprinter = not supplied                                        *
*                                                                     *
*    These values are placed in the return field, and the status byte *
*    set to zero to indicate that a selection has been made.          *
*                                                                     *
*    Return is then made to the calling program.                      *
*                                                                     *
*    If the list contains no elements, then no action is taken.       *
*                                                                     *
*    EXIT-NORMAL =                                                    *
*         Exit is via an EXEC CICS RETURN command.                    *
*         Status is set to zero if all processing completes normally. *
*                                                                     *
*    EXIT-ERROR =                                                     *
*         Exit is via an EXEC CICS RETURN command.                    *
*         Status is non-zero on entry to this module, and is left     *
*         untouched if any error occurs, hence, a non-zero return     *
*         code is passed back to the calling program.                 *
*                                                                     *
* Function 2 - autoinstall DELETE                                     *
* ------------------------------                                      *
*                                                                     *
*    Parameter list:                                                  *
*                                                                     *
*    Fullword 1 =   Function field                                    *
*      Byte 1     -   Request type(X'F1' for DELETE)                  *
*      Byte 2     -   Reserved                                        *
*      Byte 3     -   Reserved                                        *
*      Byte 4     -   Reserved                                        *
*    Fullword 2 =   Terminal id of deleted terminal                   *
*                                                                     *
*    This function gives the user the opportunity to perform          *
*    processing when an autoinstalled terminal has been deleted.      *
*                                                                     *
*    The default action of this program is to establish addressability*
*    to the parameter list, and RETURN.                               *
*                                                                     *
*    EXIT-NORMAL =                                                    *
*         Exit is via an EXEC CICS RETURN command.                    *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* ENTRY POINT = DFHZATDX                                              *
*                                                                     *
*     PURPOSE = All Functions                                         *
*                                                                     *
*     The request type is analyzed, and control passed to the         *
*     appropriate routine.                                            *
*                                                                     *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* EXTERNAL REFERENCES = None                                          *
*                                                                     *
*     ROUTINES =                                                      *
*         EXEC CICS RETURN - return to calling program                *
*                                                                     *
*                                                                     *
*     CONTROL BLOCKS =                                                *
*         See FUNCTION section for description of input parameters    *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* DESCRIPTION                                                         *
*                                                                     *
* A check is made to ensure the presence of the input parameters      *
* (passed via COMMAREA). If these do not exist, then return is made   *
* to the calling program.                                             *
*                                                                    *
* The type of request(INSTALL|DELETE) is then determined, and a      *
* branch taken to the appropriate function routine(see 'FUNCTION'    *
* above for details).                                                *
*                                                                    *
**********************************************************************
DFHEISTG DSECT ,
*
INPARMDS DSECT ,                       DEFINE INSTALL PARAMETER LIST
INPARM0  DS    F                       FUNCTION FIELD
         ORG   INPARM0                   DEFINE FUNCTION FIELD
INRQTYPE DS    XL1                       - INSTALL REQUEST TYPE
INPARM02 DS    XL1                       - RESERVED
INPARM03 DS    XL1                       - RESERVED
INPARM04 DS    XL1                       - RESERVED
         ORG   ,
INPARM1  DS    A                       POINTER TO NETNAME
INPARM2  DS    A                       POINTER TO MODEL NAME LIST
INPARM3  DS    A                       POINTER TO RETURN FIELD
INPARM4  DS    A                       POINTER TO CINIT_RU
*
DLPARMDS DSECT ,                       DEFINE DELETE PARAMETER LIST
DLPARM0  DS    F                       FUNCTION FIELD
         ORG   DLPARM0                   DEFINE FUNCTION FIELD
DLRQTYPE DS    XL1                       - DELETE REQUEST TYPE
DLPARM02 DS    XL1                       - RESERVED
DLPARM03 DS    XL1                       - RESERVED
DLPARM04 DS    XL1                       - RESERVED
         ORG   ,
DLTERMID DS    CL4                     TERMID BEING PROCESSED
*
MODLSTDS DSECT ,                       DEFINE MODEL NAME ENTRY
MODLNUM  DS    XL2                     NUMBER OF ENTRIES IN LIST
MODLNAME DS    CL8                     FIRST ENTRY IN LIST
*
OUTPRMDS DSECT ,                       DEFINE OUTPUT PARAMETERS
MODNMSEL DS    CL(L'MODLNAME)          MODEL NAME SELECTED
TRMIDSEL DS    CL4                     TERMINAL ID SELECTED
PRTTOSEL DS    CL4                     PRINTTO SELECTED
ALTPTSEL DS    CL4                     ALTPRT SELECTED
INSTATUS DS    CL1                     STATUS INFORMATION
*
NETNAMDS DSECT ,                       DEFINE NETNAME FIELD
NETNAMLN DS    XL2                     LENGTH OF NETNAME
NETNAME  DS    0X                      START OF NETNAME
*
         DFHEJECT
DFHZATDX CSECT ,
**********************************************************************
* *                  I N I T I A L I Z A T I O N                   * *
* *                  ---------------------------                   * *
* *                                                                * *
**********************************************************************
         DFHREGS ,                     EQUATE REGISTERS
         OC    EIBCALEN,EIBCALEN       ANY COMMAREA?
         BZ    RETURN                  ...NO, GET OUT
         L     R2,DFHEICAP        ADDRESS INPUT PARAMETER LIST
         USING INPARMDS,R2        BASE DSECT
         CLI   INRQTYPE,C'0'      INSTALL REQUEST?
         BNE   DELPROC1           ...NO, CHECK DELETE
         DFHEJECT
**********************************************************************
* *             I N S T A L L   P R O C E S S I N G               * *
* *             -----------------------------------               * *
* *                                                               * *
**********************************************************************
         L     R4,INPARM2         ADDRESS MODEL NAME LIST
         USING MODLSTDS,R4        BASE DSECT
         OC    MODLNUM,MODLNUM    ANY MODEL NAMES?
         BZ    RETURN             ...NO, GET OUT
         L     R5,INPARM3         ADDRESS OUTPUT PARAMETER LIST
         USING OUTPRMDS,R5        BASE DSECT
         L     R2,INPARM1         ADDRESS NETNAME INPUT FIELD
         USING NETNAMDS,R2        BASE DSECT
*
* SELECT MODEL
*
         MVC   MODNMSEL,MODLNAME  CHOOSE FIRST MODEL NAME
*
* DERIVE TERMID FROM NETNAME
*
         LA    R8,NETNAME         ADDRESS NETNAME FIELD
         LH    R6,NETNAMLN        PICK UP NETNAME LENGTH
         LA    R7,4               SET LENGTH FOR COMPARE
         CR    R6,R7              NETNAME LONGER THAN 4 CHARS?
         BNH   NETNAMSL           ...NO, TAKE FIRST N CHARS
NETSCAN1 DS    0H
* SCAN TO FIND LAST 4 NON-BLANK CHARS
         BCTR  R6,R0              DECREMENT FOR NEXT CHAR
         LA    R7,0(R6,R8)        ADDRESS NEXT CHAR
         CLI   0(R7),C' '         IS IT BLANK?
         BE    NETSCAN1           ...YES, TRY NEXT CHARACTER
         LA    R7,3               SET LENGTH FOR SUBTRACT
         AR    R8,R6              ADDRESS END OF NETNAME
         SR    R8,R7              ADDRESS LAST 4 NON-BLANK CHARS
         LA    R6,4               SET LENGTH OF 4 FOR MOVE
NETNAMSL DS    0H
* MOVE DERIVED TERMID TO RETURN FIELD. R8==>START ADDR, R6=LENGTH
         BCTR  R6,R0              SET LENGTH FOR EXECUTE
         EX    R6,TERMIDMV        SELECT TERMID
*
* SELECTIONS COMPLETE, RETURN
*
         MVI   INSTATUS,X'00'     INDICATE ALL OK
         B     RETURN             EXIT PROGRAM
*
TERMIDMV MVC   TRMIDSEL(0),0(R8)  EXECUTED MOVE FOR TERMID SELECTION
*
         DROP  R2
         DROP  R4
         DROP  R5
*
```

```
         DFHEJECT
***********************************************************************
* *            D E L E T E   P R O C E S S I N G                    * *
* *            ----------------------------------                   * *
* *                                                                 * *
***********************************************************************
DELPROC1 DS    0H
         USING DLPARMDS,R2         BASE DSECT
         CLI   DLRQTYPE,C'1'       DELETE REQUEST?
         BNE   RETURN              ...NO, ERROR, EXIT
* ==> PUT DELETE CODE HERE
         B     RETURN              EXIT PROGRAM
*
         DFHEJECT
RETURN   DS    0H
         EXEC CICS RETURN ,
         END   DFHZATDX
```

COBOL version of the example program

The prolog of this program has been omitted. It is based on the assembler prolog.

```
  IDENTIFICATION DIVISION.
  PROGRAM-ID. DFHZCTDX.

  ENVIRONMENT DIVISION.

  DATA DIVISION.
  WORKING-STORAGE SECTION.
*
* CODES SUPPLIED BY COMMAREA:
*
  77  INSTALL-CODE PIC X(1) VALUE IS '0'.
  77  DELETE-CODE PIC X(1) VALUE IS '1'.
*
* STRUCTURE TO ALLOW THE LAST FOUR CHARACTERS TO BE USED AS
* THE NETNAME.
*
  01  NETNAME-BITS.
      02  FIRST-HALF.
          03 CHR-1 PIC X(1).
          03 CHR-2 PIC X(1).
          03 CHR-3 PIC X(1).
          03 CHR-4 PIC X(1).
      02  SECOND-HALF.
          03 CHR-5 PIC X(1).
          03 CHR-6 PIC X(1).
          03 CHR-7 PIC X(1).
          03 CHR-8 PIC X(1).
*
* TERMINAL IDENTIFIER IS BUILT HERE BEFORE BEING PLACED IN THE
* RETURN FIELD.
*
  01  TERM-IDNT.
      02 TERM-CHR1 PIC X(1).
      02 TERM-CHR2 PIC X(1).
      02 TERM-CHR3 PIC X(1).
      02 TERM-CHR4 PIC X(1).
  LINKAGE SECTION.
*

* COMMAREA FORMAT:
*   FULLWORD 1 - FUNCTION FIELD
*                BYTE 1 - REQUEST TYPE
*                BYTE 2 - RESERVED
*                BYTE 3 - RESERVED
*                BYTE 4 - RESERVED.
*   FULLWORD 2 - POINTER TO NETNAME.
*   FULLWORD 3 - POINTER TO MODEL NAME LIST.
*   FULLWORD 4 - POINTER TO RETURN FIELD.
*   FULLWORD 5 - POINTER TO CINIT_RU.
*
  01  DFHCOMMAREA.
      02  FUNCTION-FIELD.
          03  REQUEST-TYPE PIC X(1).
          03  REST  PIC X(3).
      02  NET-PTR PIC S9(8) COMP.
      02  MOD-PTR PIC S9(8) COMP.
      02  RET-FLD-PTR PIC S9(8) COMP.
      02  CINRU-PTR PIC S9(8) COMP.
*
* PARMLIST ENABLES THE DATA AREAS POINTED TO BY THE COMMAREA
* TO BE ACCESSED
*
  01  PARMLIST.
      02  FILLER PIC S9(8) COMP.
      02  NETNAME-PTR PIC S9(8) COMP.
      02  MODENAME-PTR PIC S9(8) COMP.
      02  RETFLD-PTR PIC S9(8) COMP.
      02  CINITRU-PTR PIC S9(8) COMP.
*
*
*
  01  NETNAME.
      02  NETNAME-LENGTH PIC S9(2) COMP.
      02  NETNAME-NAME PIC X(8).
*
* MODELNAME IS NOT USED IN THIS PROGRAM AS THE IBM SUPPLIED
* AUTOINSTALL MODELS ARE NOT BEING USED. MODELNAME IS A LIST
* OF AUTOINSTALL MODELS SO IF MORE THAN THE FIRST MODEL IS TO
* BE USED A LONGER DEFINITION WILL BE REQUIRED:
*
* 01  MODELNAME.
*     02  MODELNAME-LENGTH PIC X(2).
*     02  MODELNAME-NAME1 PIC X(8).
*     02  MODELNAME-NAME2 PIC X(8).
*     02  MODELNAME-NAME3 PIC X(8).
*     02  MODELNAME-NAME4 PIC X(8).
*         ...ETC.
*
```

```
  01  MODELNAME.
      02  NO-MODELS PIC S9(2) COMP.
      02  MODELNAME-NAME PIC X(8).
*
* RETURN-FIELD WILL CONTAIN THE DATA TO BE RETURNED TO CICS
*
  01  RETURN-FIELD.
      02  MOD-NAME PIC X(8).
      02  INSTANCE-NAME PIC X(4).
      02  PRINTTO PIC X(4).
      02  ALTPRT PIC X(4).
      02  RET-STATUS PIC X(1).
  01  CINIT-AREA.
      02  CINITRU-LENGTH PIC S9(4).
      02  CINITRU PIC X(256).
 PROCEDURE DIVISION.
*
* SET UP ADDRESSABILITY TO THE PARMLIST.
*
     SERVICE RELOAD PARMLIST.
*
* CHECK THAT WE HAVE A COMMAREA AND SOME MODELS TO USE, IF NOT
* THEN EXIT
*
     IF EIBCALEN EQUAL 0 THEN PERFORM RETURN-LINE.
     IF NO-MODELS EQUAL 0 THEN PERFORM RETURN-LINE.
*
* SET UP ADDRESSABILITY TO THE COMMAREA - MOVE POINTERS AND
* THEN RELOAD THE DATA.
*
     MOVE NET-PTR TO NETNAME-PTR.
     SERVICE RELOAD NETNAME.
*
     MOVE MOD-PTR TO MODENAME-PTR.
     SERVICE RELOAD MODELNAME.
*
     MOVE RET-FLD-PTR TO RETFLD-PTR.
     SERVICE RELOAD RETURN-FIELD.
*
     MOVE CINRU-PTR TO CINITRU-PTR.
     SERVICE RELOAD CINIT-AREA.
*
* EXECUTE THE APPROPRIATE PARAGRAPH FOR INSTALL OR DELETE:
*
     IF REQUEST-TYPE EQUAL INSTALL-CODE THEN
        PERFORM INSTALL-PARAGRAPH.
*
* IF THE REQUEST WAS AN INSTALL REQUEST THEN THE NEXT TEST
* WILL FAIL ANYWAY, IE. FANCY LOGIC NOT REQUIRED!
*
     IF REQUEST-TYPE EQUAL DELETE-CODE THEN
        PERFORM DELETE-PARAGRAPH.
*
* RETURN TO CICS.
*
 RETURN-LINE.
```

```
     EXEC CICS RETURN END-EXEC.
*
*
 INSTALL-PARAGRAPH.
*
* MOVE THE NETNAME SO THAT IT CAN BE DEALT WITH ON A CHARACTER TO
* CHARACTER BASIS.
*
     MOVE NETNAME-NAME TO NETNAME-BITS.
*
* RESET NETNAME LENGTH IF THERE ARE TRAILING SPACES.
*
     IF NETNAME-LENGTH = 8 AND CHR-8 = ' ' THEN
        MOVE 7 TO NETNAME-LENGTH.
     IF NETNAME-LENGTH = 7 AND CHR-7 = ' ' THEN
        MOVE 6 TO NETNAME-LENGTH.
     IF NETNAME-LENGTH = 6 AND CHR-6 = ' ' THEN
        MOVE 5 TO NETNAME-LENGTH.
     IF NETNAME-LENGTH = 5 AND CHR-5 = ' ' THEN
        MOVE 4 TO NETNAME-LENGTH.
*
* MAKE UP TERMINAL IDENTIFIER FROM NETNAME.
*
     IF NETNAME-LENGTH < 5 THEN MOVE FIRST-HALF TO TERM-IDNT.
     IF NETNAME-LENGTH = 5 THEN
        MOVE CHR-2 TO TERM-CHR1
        MOVE CHR-3 TO TERM-CHR2
        MOVE CHR-4 TO TERM-CHR3
        MOVE CHR-5 TO TERM-CHR4.
     IF NETNAME-LENGTH = 6 THEN
        MOVE CHR-3 TO TERM-CHR1
        MOVE CHR-4 TO TERM-CHR2
        MOVE CHR-5 TO TERM-CHR3
        MOVE CHR-6 TO TERM-CHR4.
     IF NETNAME-LENGTH = 7 THEN
        MOVE CHR-4 TO TERM-CHR1
        MOVE CHR-5 TO TERM-CHR2
        MOVE CHR-6 TO TERM-CHR3
        MOVE CHR-7 TO TERM-CHR4.
     IF NETNAME-LENGTH = 8 THEN MOVE SECOND-HALF TO TERM-IDNT.
*
* PLACE TERM-IDNT IN RETURN FIELD
*
     MOVE TERM-IDNT TO INSTANCE-NAME.
*
* SELECT THE MODEL FROM THE LIST SUPPLIED (THE FIRST MODEL IS
* SELECTED).
*
*
     MOVE MODELNAME-NAME TO MOD-NAME.
*
* PUT RETURN CODE 0 INTO THE STATUS BYTE.
*
     MOVE LOW-VALUES TO RET-STATUS.
*
*
```

```
DELETE-PARAGRAPH.
*
* DELETE CODE IS PLACED HERE.
*
* RETURN TO CICS
*
    EXEC CICS RETURN END-EXEC.
* END
    STOP RUN.
```

PL/I version of the example program

The prolog of this program has been omitted. It is based on the assembler prolog.

```
DFHZPTDX: PROC (DFHCOM)
         OPTIONS(MAIN,REENTRANT) REORDER;
        DCL   DFHCOM                   POINTER;
        DCL 1 COMM_1                   BASED(DFHCOM),
              2 DFHFUNC                CHAR(1),
                       /* '0' FOR INSTALL                          */
                       /* '1' FOR DELETE                           */
              2 FILL                   CHAR(3),
              2 DFHNET                 POINTER,
                       /* ADDRESS OF NETNAME AREA                  */
              2 DFHMDL                 POINTER,
                       /* ADDRESS OF MODELS(S) AREA                */
              2 DFHRET                 POINTER,
                       /* ADDRESS OF RETURNED DATA AREA            */
              2 DFHCINIT               POINTER;
                       /* ADDRESS OF VTAM CINIT RU                 */
        DCL  1 NET                     BASED(COMM_1.DFHNET),
              2 NETNAME                CHAR(8) VARYING;
                       /* NETNAME OF TERMINAL TRYING CONNECTION    */
        DCL 1 MODEL                    BASED (COMM_1.DFHMDL),
              2 NO_MODELS              FIXED BIN(15),
                       /* NUMBER OF MODELS IN LIST                 */
              2 MODEL_NAMES(LN REFER (NO_MODELS)) CHAR(8);
                       /* MODEL NAMES RETURNED BY CICS             */
        DCL   1 RETURN                 BASED(COMM_1.DFHRET),
              2 AUTO_NAME              CHAR(8),
                       /* MODEL NAME RETURNED BY EXIT TO CICS      */
              2 NEW_TERMID             CHAR(4),
                       /* TERMINAL NAME YOU HAVE DECIDED ON        */
              2 NEW_PRINTER            CHAR(4),
              2 ALT_PRINTER            CHAR(4),
                       /* PRINTER NAMES YOU HAVE DECIDED ON        */
              2 AUTO_STAT              CHAR(1);
                       /* SET TO X'00' TO ALLOW LOGON              */
        DCL   1 COMM_2                 BASED(DFHCOM),
              2 DFHFUNC                CHAR(1),
              2 FILL                   CHAR(3),
              2 OLD_TERMID             CHAR(4);
                       /* SET TO TERMID TO DELETE                  */
        DCL 1 CINIT                    BASED(COMM_1.DFHCINIT),
              2 CINIT_LEN              FIXED BIN(15),
                    /* LENGTH OF CINIT RU                                    */
              2 CINIT_RU                CHAR(256);
                    /* CINIT RU ITSELF                                       */
        DCL   SAVE_NET                  CHAR(8);
                    /* TEMP SAVE AREA FOR NETNAME                            */
        DCL ( LOW,SUBSTR,CSTG)          BUILTIN;
        DCL   I                         FIXED BIN(15) INIT (0);
        /* THREE MAIN AREAS OF PROCESSING
           1 = INSTALL A TERMINAL
           2 = DELETE A TERMINAL
           3 = PRODUCE INFORMATIONAL MESSAGES
        FOR AN INSTALL REQUEST A COMMAREA IS PASSED (AS FOLLOWS)
        FULLWORD 1 -  FUNCTION FIELD
          BYTE 1 = CHARACTER '0'
          BYTES 2-4 RESERVED
        FULLWORD 2 ==> NETNAME AREA
                         BYTES 1-2 LENGTH IN BINARY OF NETNAME
                         BYTES 3-* NETNAME ITSELF
        FULLWORD 3 ==> MODEL NAME LIST
                         BYTES 1-2 NUMBER OF MODELS IN LIST
                         BYTES 3-*EIGHT BYTE MODEL NAMES
        FULLWORD 4 ==> RETURNED DATA TO CICS
                         BYTES 1-8 AUTO INSTALL MODEL NAME TO USE
                         BYTES 9-12 NEW TERMINAL IDENTITY (UNIQUE)
                         BYTES 13-16 RELATED PRINTER IDENTITY
                         BYTES 17-20 ALTERNATE PRINTER IDENTITY
                         BYTE  21    RETURN CODE
        FULLWORD 5 ==> VTAM CINIT RU
                         BYTES 1-2 BINARY LENGTH OF CINIT RU
                         BYTES 3-* CINIT RU ITSELF
        THIS EXIT WILL ATTEMPT TO MAKE A TERMID FROM THE NETNAME
        AND TAKE THE FIRST MODEL FROM THE SET OF MODELS DEFINED
        (FROM THE CSD) , SET UP THE PRINTER
        DETAILS AND SET A STATUS CODE OF X00,
        ANY STATUS CODE OTHER THAN X00 CAUSES CICS TO REJECT THE
        AUTOINSTALL ATTEMPT.

  FOR A DELETE REQUEST A COMMAREA IS AGAIN PASSED BUT WITH
  THE FOLLOWING FORMAT
        FULLWORD 1 FUNCTION REQUEST
                         BYTE 1 CHARACTER '1'
                         BYTES 2-4 RESERVED
        FULLWORD 2 CHARACTER 4 IDENTITY OF DELETED TERMINAL
                                                                             */
        IF    EIBCALEN = 0                              /* IF NO COMMAREA */
        THEN DO;                                            /*THEN IGNORE */
EXIT:                                               /* GENERAL EXIT POINT */
              EXEC CICS RETURN;
             END;
                                    /* CHECK WHAT FUNCTION IS REQUESTED */
                                    /* AND CALL THE APPROPRIATE ROUTINE */
 SELECT (COMM_1.DFHFUNC);
  WHEN ('0') CALL INSTALL;
  WHEN ('1') CALL DELETE;
  OTHERWISE  CALL DUMP;
 END;
 INSTALL: PROC REORDER;                            /* INSTALL PROCESS   */
                       /* CHECK IF ANY MODELS PRESENTED BY CICS   */
        IF NO_MODELS = 0                          /* IF NO MODELS   */
        THEN GO TO EXIT;                             /* THEN EXIT   */
         AUTO_NAME = MODEL_NAMES(1);
                /* MOVE THE NETNAME TO AN 8 BYTE AREA TO WORK ON   */
         SAVE_NET = NETNAME;
         DO I = 8 TO 1 BY -1;           /* FIND LAST FOUR NON BLANK   */
          IF SUBSTR(SAVE_NET,I,1) ¬= ' '
          THEN LEAVE;
         END;                            /* IF LENGTH LESS THAN 4 USE   */
                                         /* NETNAME, OTHERWISE USE      */
                                         /* LAST FOUR NON BLANK         */
         IF I < 4 THEN NEW_TERMID = NETNAME;
                   ELSE NEW_TERMID = SUBSTR(SAVE_NET,I-3,4);
          NEW_PRINTER, ALT_PRINTER = '        ';
                                         /*NOT INTERESTED IN PRINTERS   */
          AUTO_STAT = LOW(1);
                                  /* SET STAT FIELD TO X'00' TO ALLOW
                                               LOGON TO BE PROCESSED   */
          GO TO EXIT;
END INSTALL;
               /* FOR A DELETE REQUEST THERE IS VERY LITTLE TO DO */
DELETE: PROC REORDER;
                                               /* ADD DELETE CODE HERE */
      EXEC CICS RETURN;
 END DELETE;
DUMP:    PROC REORDER;          /* PRODUCE A DUMP IF INVALID REQUEST   */
         EXEC CICS DUMP TASK DUMPCODE('AUTO');
         EXEC CICS RETURN;
 END DUMP;
 END DFHZPTDX;
```

Customizing the example program

Here are three pieces of code that customize the example user program.

Assembler

This example, in assembler, limits logon to netnames L77A and L77B. The model names utilized are known in advance. A logon request from any other terminal, or a request for a model which cannot be found, will be rejected.

```
*    REGISTER CONVENTIONS =                                            *
*      R0  used by DFHEICAL expansion                                  *
*      R1  -------"-------"------"----                                 *
*      R2  Base for Input parameters                                   *
*      R3  Base for code addressability                                *
*      R4  Base for model name list                                    *
*      R5  Base for output parameter list                              *
*      R6  Work register                                               *
*      R7  -----"-------                                               *
*      R8  -----"-------                                               *
*      R9  free                                                        *
*      R10 Internal subroutine linkage return                          *
*      R11 Base for EIB                                                *
*      R12 free                                                        *
*      R13 Base for dynamic storage                                        *
*      R14 used by DFHEICAL expansion                                      *
*      R15 -------"-------"------"----                                     *
                                                                           *

* SELECT MODEL
*
         LH    R6,TABLEN          NUMBER OF VALID NETNAMES
         LA    R7,TABLE           ADDRESS THE TABLE
*
LOOP1    CLC   NETNAME(4),0(R7)   IS THIS NETNAME IN TABLE
         BE    VALIDT
*
         LA    R7,16(R7)          NEXT TABLE ENTRY
         BCT   R6,LOOP1
*
*        NOW WE KNOW ITS NOT A VALID NETNAME
*        SIMPLY RETURN AND THE LOGON IS REJECTED
*
         B     RETURN
*
*                                 R7 NOW POINTS TO MODELNAME
VALIDT   LH    R6,MODLNUM         REQUIRED, NOW SEE IF IT WAS
         LA    R8,MODLNAME        PRESENTED TO THE EXIT
*
LOOP2    CLC   8(8,R7),0(R8)      IS THIS MODELNAME HERE
         BE    VALIDM
*
         LA    R8,L'MODLNAME(R8)  NEXT MODLNAME
         BCT   R6,LOOP2
*
*        NOW WE KNOW THE REQUIRED MODELNAME WAS NOT PRESENTED
*        TO THIS EXIT BY CICS, A RETURN REJECTS THE LOGON
*
         B     RETURN
*
*        AT THIS POINT THE MODELNAME WAS FOUND IN THOSE PRESENTED
*        IT IS GIVEN TO CICS AND THE NEW TERMID  WILL BE
*        THE NETNAME
*
VALIDM   MVC   MODNMSEL,0(R8)     R8 WAS LEFT POINTING AT MODELNAME
         MVC   TRMIDSEL,NETNAME   USE NETNAME FOR TERMID(4 CHARS)
*
*
* SELECTIONS COMPLETE, RETURN
*
         MVI   INSTATUS,X'00'     INDICATE ALL OK
         B     RETURN             EXIT PROGRAM
*

*        TABLE OF NETNAMES ALLOWED TO LOGON AND THE MODEL NAME
*        NECESSARY FOR THE LOGON TO BE SUCCESSFUL
*
*        FORMAT OF TABLE :
*            BYTES 1  TO  8     NETNAME ALLOWED TO LOGON
*                  9  TO  16    MODEL REQUIRED FOR NETNAME
*
         DS     0D
TABLE    DC     CL8'L77A',CL8'3270064'
         DC     CL8'L77B',CL8'3270065'
TABLEN   DC     Y((*-TABLE)/16)
*
```

COBOL

The second example, in COBOL, redefines the NETNAME, so that the last four characters are used to select a more suitable model than that selected in the example user program.

```
               .
               .
               .
*
* Redefine the NETNAME so that the last 4 characters (of 7)
* can be used to select the AUTOINSTALL model to be used.
*
* The netnames to be supplied are known to be of the form:
*
*     HVMXNNN
*
* HVM is the prefix
* X   is the system name
* NNN is the address of the terminal
*
  01  NETNAME-BITS.
      02  FIRST-CHRS PIC X(3).
      02  NEXT-CHRS.
          03 NODE-LETTER PIC X(1).
          03 NODE-ADDRESS PIC X(3).
      02  LAST-CHR PIC X(1).
               .
               .
               .
 PROCEDURE DIVISION.
               .
               .
               .
               .
*
* Select the AUTOINSTALL model to be used according to the
* NODE LETTER (see above).  The models to be used are user
* defined.
*
* (It is assumed that the NETNAME supplied in the COMMAREA by CICS
*  has been MOVEd to NETNAME-BITS).
*
* If the node letter is C then use model AUTO2
* If the terminal NETNAME is HVMC289 (a special case) then use
* model AUTO1.
* Otherwise (node letters A,B,D...) use model AUTO3.
*
```

```
IF NODE-LETTER = 'C' THEN MOVE 'AUTO2' TO MOD-NAME.
IF NEXT-CHRS = 'C289' THEN MOVE 'AUTO1' TO MOD-NAME.
IF NODE-LETTER = 'A' THEN MOVE 'AUTO3' TO MOD-NAME.
IF NODE-LETTER = 'B' THEN MOVE 'AUTO3' TO MOD-NAME.
IF NODE-LETTER = 'D' THEN MOVE 'AUTO3' TO MOD-NAME.
          .
          .
          .
          .
```

PL/I

The third example, in PL/I, extracts information from the VTAM CINIT RU, which carries the bind image. Part of this information is the screen presentation services information, such as the default screen size and alternate screen size. The alternate screen size is used to determine the model of terminal that is requesting logon. The presented models are searched for a match, and if there is no match the first model from those presented is used.

```
DCL   SAVE_CINIT                CHAR(256);
                               /* TEMP SAVE AREA FOR CINIT RU */

DCL 1 SCRNSZ                    BASED(ADDR(SAVE_CINIT)),
      2 SPARE                    CHAR(31),
                      /* BYPASS FIRST PART OF CINIT AND REACH */
                          /* INTO BIND IMAGE CARRIED IN CINIT */
      2 DHGT                     BIT(8),
                     /* SCREEN DEFAULT HEIGHT IN BIND PS AREA */
      2 DWID                     BIT(8),
                      /* SCREEN DEFAULT WIDTH IN BIND PS AREA */
      2 AHGT                     BIT(8),
                   /* SCREEN ALTERNATE HEIGHT IN BIND PS AREA */
      2 AWID                     BIT(8);
                    /* SCREEN ALTERNATE WIDTH IN BIND PS AREA */

DCL   NAME                      CHAR(2);
                       /* USED TO WORK UP A SCREEN MODEL TYPE */
DCL   TERMID                    PIC'9999' INIT(1) STATIC;
                           /* USED TO WORK UP A UNIQUE TERMID */
DCL   ENQ                       CHAR(8) INIT('AUTOPRG');
                 /* USED TO PREVENT MULTIPLE ACCESS TO TERMID */

  /* CLEAR THE CINIT SAVE AREA AND MOVE IN THE VTAM CINIT RU */
        /* THIS IS USEFUL IF YOU FAIL TO RECOGNIZE THE MODEL */
        /* OF TERMINAL, PROVIDE A DUMP AND ANALYSE THIS DATA */

SAVE_CINIT = LOW(256);
SUBSTR(SAVE_CINIT,1,CINIT_LEN) = SUBSTR(CINIT_RU,1,CINIT_LEN);

/* NOW ACCESS THE SCREEN PS AREA IN THE PORTION OF THE BIND
   IMAGE PRESENTED IN THE CINIT RU */
/* USING THE SCREEN ALTERNATE HEIGHT AS A GUIDE TO THE MODEL
   OF TERMINAL ATTEMPTING LOGON, IF THIS CANNOT BE DETERMINED
   THEN DEFAULT TO THE FIRST MODEL IN THE TABLE */

SELECT (AHGT);                /* NOW GET SCRN ALTERNATE HEIGHT *
```

```
         WHEN (12)  NAME = 'M1';        /* MODEL 1 */
         WHEN (32)  NAME = 'M3';        /*       3 */
         WHEN (43)  NAME = 'M4';        /*       4 */
         WHEN (27)  NAME = 'M5';        /*       5 */

         OTHERWISE  NAME = 'M2';        /*       2 */

        END;

        /* SEARCH THE MODEL ENTRIES FOR A MATCHING ENTRY       */
        /* THE CRITERION HERE IS THAT A MODEL DEFINITION SHOULD*/
        /* CONTAIN THE CHARS M2 FOR A MODEL 2 ETC              */
        /* E.G. L3270M2, L3270M5                               */
        /*     TERMM2,  TERMM5                                 */

        DO I = 1 TO NO_MODELS;
         IF INDEX(MODEL_NAMES(I),NAME)
         THEN GOTO FOUND_MODEL;
        END;

NO_MODEL: /* MATCHING ENTRY WAS NOT FOUND, DEFAULT TO FIRST MODEL*/
        AUTO_NAME = MODEL_NAMES(1);
        GO TO MODEL_EXIT;

FOUND_MODEL: /* MOVE THE SELECTED MODEL NAME TO THE RETURN AREA */
         AUTO_NAME = MODEL_NAMES(I);

MODEL_EXIT: /* ENQ TO STOP MULTIPLE UPDATES OF COUNTER      */
           /* A SIMPLE COUNTER IS USED TO GENERATE UNIQUE  */
           /* TERMINAL IDENTITIES, SO CONCURRENT ACCESS IS */
           /* DENIED TO THIS COUNTER TO ENSURE NO TWO GET  */
           /* THE SAME IDENTITY OR UPDATE THE COUNTER      */

  /* TO USE THIS METHOD THE PROGRAM MUST BE DEFINED AS RESIDENT */

         EXEC CICS ENQ RESOURCE(ENQ);

         NEW_TERMID = TERMID; /* SET NEW_TERMID TO COUNT VALUE  */
         TERMID = TERMID + 1; /* INCREASE THE COUNT VALUE BY 1  */
         IF TERMID = 9999 THEN TERMID = 1;     /* RESET IF TOO LARGE*/

         EXEC CICS DEQ RESOURCE(ENQ);

NAME_EXIT:
         NEW_PRINTER, ALT_PRINTER = LOW(4);
                                        /*NOT INTERESTED IN PRINTERS */

         AUTO_STAT = LOW(1);
                              /* SET STAT FIELD TO X"00" TO ALLOW
                                 LOGON TO BE PROCESSED */
         GO TO EXIT;
END INSTALL;
```

Chapter 4.9. Exits for "terminal-not-known" condition

This chapter starts with a brief summary of the circumstances that give rise to the "terminal-not-known" condition. The body of the chapter describes the exits that are provided to help you to deal with the condition. The chapter concludes with a listing of the supplied sample exit program.

The terminal-not-known condition

The terminal-not-known condition can occur when intercommunicating CICS regions use both SHIPPABLE terminal definitions and automatic transaction initiation (ATI). The condition is especially likely to arise if autoinstall is used.

SHIPPABLE attribute
Terminals defined with the SHIPPABLE attribute in a terminal-owning region (TOR) do not need a definition in a connected application-owning region (AOR). If necessary to support transaction routing, CICS ships a copy of the definition from the TOR to the AOR. For full information, see the *CICS/MVS Resource Definition (Online)* manual.

Automatic transaction initiation (ATI)
ATI occurs when an internally generated request leads to the initiation of a transaction. For, example:

An application issues an EXEC CICS START command [1], or
The transient data trigger level is reached.

Two CICS modules handle ATI requests:

The interval control program processes a START command, checks that the terminal is known in the local system, and (when any START time interval elapses) calls the terminal allocation program.

The terminal allocation program is called by the interval control program or by the transient data triggering mechanism, and checks that the terminal is known in the local system. If the requested terminal is remote, the terminal allocation program ships an ATI request to the remote system, which initiates transaction routing back to the local system.

For full information on ATI, see the *CICS/MVS Intercommunication Guide*.

Terminal-not-known condition
The **terminal-not-known** condition arises when an ATI request is made for a terminal not known in the region. An ATI request can occur in the AOR for a **SHIPPABLE** terminal before any transaction routing has taken place for the terminal, and so before the definition of the terminal can have been shipped from the TOR to the AOR.

[1] In this chapter, all statements about START commands apply equally to DFHIC TYPE=INITIATE and DFHIC TYPE=PUT macros, except that the macros cannot be function shipped.

If the terminal-not-known condition occurs, both the interval control program and the terminal allocation program reject the transaction-initiation request as **TERMIDERR**.

The exits

To deal with the terminal-not-known condition, CICS provides global user exits in the interval control and terminal allocation programs.

The exits are **XICTENF** in the interval control program and **XALTENF** in the terminal allocation program

CICS drives the XICTENF exit only when the terminal-not-known condition occurs. CICS drives the XALTENF exit only when the terminal-not-known condition occurs *and* the terminal allocation program has been invoked by the transient data trigger level or the interval control program.

The exit program must indicate whether the terminal exists on another system and, if so, which one. CICS passes data to the exit program to help it make its decision. You can use the same exit program at both exit points. CICS supplies a sample exit program, DFHXTENF (see Figure 23 on page 282), that can be used at both exits and that can deal unchanged with some typical situations. To define the exit program to CICS, use the CEDA DEFINE PROGRAM transaction.

The exits are designed to deal with terminal-not-known conditions that occur in CICS systems other than the TOR. For a TOR/AOR pair, enable the exits in the the AOR. The exits cannot deal with a terminal-not-known condition in the TOR and should not normally be enabled there. If more than one TOR exists, you may need to enable the exits in each TOR to deal with requests for terminals owned by other TORs. In this case, the exit program must recognize terminals that should be owned by this system and reject the request (return code 0). Although the exit provides as much data as possible, the logic of your program depends entirely on your system design. A simple solution to the most complex case would be to make the name of each terminal reflect the NETNAME or SYSID of its owning region.

Note: If a CICS APPLID is used simultaneously by two connections (for example, LU62 and MRO connections exist at the same time between two CICS systems), both connections have the same NETNAME. If the exit program returns a NETNAME in this case, CICS may locate the wrong connection, with resulting failure of transaction initiation. If more than one connection can use the APPLID, the exit program should return the SYSID, which uniquely identifies the connection.

Data passed to exit

For both exits, the address of a parameter list is in the register 1 field (offset = 24 bytes) in the register save area addressed by UEPHMSA. Table 2 on page 279 shows the contents of the parameter list. The supplied sample exit program (Figure 23) contains a DSECT, XTEPARMS, which describes the parameter list.

Data returned by exit

The exit program must set a return code in register 15 as follows:

```
0       terminal does not exist
4       NETNAME returned
8       SYSID returned.
```

For return codes 4 and 8, the program must place the NETNAME or SYSID in the fields at offsets 28 and 36 respectively of the parameter list (see Table 2).

Condition arising during initialization

The terminal-not-known condition can arise during restart when the terminal allocation program is processing transient data queues and expired ICEs. This is before post-initialization processing, the earliest time that exits can be enabled by user code. To cover this situation, the SIT operand and override parameter, ALEXIT, names a program that is enabled for the XALTENF exit during initialization. The ALEXIT specification enables the program for initialization only — you must use an ENABLE command to enable a program for the XALTENF exit for subsequent CICS processing (see "Enabling and disabling an exit program" on page 296).

Parameter list

Table 2 (Page 1 of 3). Parameter list for the XALTENF and XICTENF exits

OFFSET	LENGTH	CONTENTS
0	2	**On entry to exit program** Type of request. **C'SD'** START command with data. **C'S'** START command without data. **C'QD'** Transient data trigger level reached.
2	1	**On entry to exit program** Indication for a START command whether the task issuing the command was started by transaction routing. **C'Y'** A START command was being processed and the task was being transaction routed. **C'N'** A START command was not being processed *or* the task was not being transaction routed.
3	1	**On entry to exit program** Indication for a START command whether the command was function shipped. **C'Y'** A START command was being processed and the START was function shipped. **C'N'** A START command was not being processed *or* a START was being processed but it was not function shipped.

Table 2 (Page 2 of 3). Parameter list for the XALTENF and XICTENF exits

OFFSET	LENGTH	CONTENTS
4	4	**On entry to exit program** Name of transaction to be run.
8	4	**On entry to exit program** Name of terminal the transaction should run on. (If a transient data trigger level was reached and the DCT entry specified DESTFAC= (SYSTEM,sysidnt), this would contain a SYSID.)
12	4	**On entry to exit program** For START commands, the name of the current terminal if the command was transaction routed, or the name of the session if the command was function shipped. For other START commands and for transient data trigger events, blanks.
16	8	**On entry to exit program** For function-shipped START commands - the netname of the last system the request came from. For START commands issued in this system by a task being Transaction Routed to, the netname of the last system from which the task was routed. For other START command situations and for transient data trigger level events, blanks.
24	4	**On entry to exit program** If the field at offset 16 contains a netname, the corresponding SYSID. If the field at offset 16 does not contain a netname, blanks.
28 **Output field**	8	**On entry to exit program** The contents of the field at offset 16. *On exit from exit program if setting a return code of 4.* The netname of the system the ATI request should be sent to.

Table 2 (Page 3 of 3). Parameter list for the XALTENF and XICTENF exits		
OFFSET	**LENGTH**	**CONTENTS**
36 **Output field**	4	**On entry to exit program** The contents of the field at offset 24. *On exit from exit program if setting a return code of 8.* The sysid of the system the ATI request should be sent to.

Sample program

One program can be used for both exits or a separate program can be written for each. Figure 23 shows the supplied sample program DFHXTENF, which can be used for both exits. DFHXTENF rejects transient data requests, because the action in this case is very much installation-dependent.

```
         TITLE 'CUSTOMER INFORMATION CONTROL - SAMPLE XICTENF/XALTENF  *
               GLOBAL USER EXIT PROGRAM'
DFHXTENF  CSECT
***********************************************************************
* MODULE NAME = DFHXTENF                                              *
* DESCRIPTIVE NAME = C.I.C.S./VS Sample XICTENF/XALTENF global exit.  *
* STATUS = 2.1.0                                                      *
* FUNCTION =                                                          *
*        DFHXTENF acts as an exit program for the XALTENF and         *
*        XICTENF exits which deal with 'terminal not known'           *
*        conditions. Its purpose is to tell CICS whether the          *
*        terminal exists on another system and, if so, which one.     *
*        It is called by the User Exit Handler but only if required   *
*        and only if the user has enabled it as an exit program       *
*        for the XALTENF exit in DFHALP or XICTENF exit in DFHICP.    *
*        When a START command involving a terminal is issued,         *
*        DFHICP checks that the terminal exists. If the locate        *
*        fails to find it, DFHICP drives the XICTENF exit.            *
*        When DFHALP is asked by DFHICP or Transient Data to          *
*        start an ATI task on a given terminal, it checks that        *
*        the terminal exists. If the locate fails to find it,         *
*        DFHALP drives the XALTENF exit.                              *
*        DFHALP and DFHICP pass a common set of parameters to their   *
*        respective exits to help the exit program make its           *
*        decision. The parameter list address is at offset decimal    *
*        24 from the location addressed by UEPHMSA. The parameter     *
*        list contents are described by the XTEPARMS DSECT below.     *
*                                                                     *
*        DFHXTENF does the following:-                                *
*                                                                     *
*           It decides whether it is dealing with a START request     *
*        or a Transient Data request.                                 *
*           For START requests with a netname passed in, it returns   *
*        the same netname and a return code of 4 (terminal known).    *
*        For START requests with no netname, it constructs a netname  *
*        by taking the first character of the terminal name and       *
*        appending it to the characters 'CICS'.                       *
*           For Transient Data requests it rejects the request.       *
*                                                                     *
*                                                                     *
```

Figure 23 (Part 1 of 5). Sample program for XALTENF and XICTENF exits

```
* NOTES :                                                          *
*    DEPENDENCIES = S/370                                          *
*        DFHXTENF must be defined as a PPT program.                *
*    RESTRICTIONS = None                                           *
*    REGISTER CONVENTIONS = See the code                           *
*    MODULE TYPE = Executable                                      *
*    PROCESSOR = Assembler                                         *
*    ATTRIBUTES = Read only, Re-entrant                            *
* ENTRY POINT = DFHXTENA                                           *
*                                                                  *
*     PURPOSE = see above                                          *
*                                                                  *
*     LINKAGE = Called by the User Exit Handler                    *
*                                                                  *
*     INPUT =                                                      *
*        On entry register 1 addresses a parameter list described  *
*        by the DFHUEPAR DSECT. This in turn contains the address  *
*        of a parameter list generated by DFHALP or DFHICP. The    *
*        contents of the second parameter list are described in    *
*        Table 2 on page 279                                        *
*     OUTPUT =                                                     *
*        A return code is placed in register 15. If the return     *
*        is 4 ('terminal exists'), a netname is passed back in the *
*        second parameter list. If the return code is 8 ('terminal *
*        exists'), a sysid is passed back in the second parameter  *
*        list.                                                     *

*     EXIT-NORMAL =                                                *
*        DFHXTENF returns to the address that was in register 14   *
*        when it was called. Possible return codes in register 15  *
*        are:-                                                     *
*          0 - terminal does not exist                             *
*          4 - terminal exists, netname returned in parameter list *
*          8 - terminal exists, sysid returned in parameter list   *
*                                                                  *
*     EXIT-ERROR = none                                            *
*                                                                  *
*----------------------------------------------------------------- *
*                                                                  *
* EXTERNAL REFERENCES = none                                       *
*                                                                  *
*----------------------------------------------------------------- *
*                                                                  *
* DESCRIPTION - see above                                          *
*                                                                  *
*----------------------------------------------------------------- *
         DFHEJECT
```

Figure 23 (Part 2 of 5). Sample program for XALTENF and XICTENF exits

```
***********************************************************************
*     Parameters passed to the XALTENF and XICTENF exits              *
***********************************************************************
XTEPARMS DSECT                      exit parameter list           @D2A
XTEEVENT DS    CL2                  reason for exit being driven
*                                   C'QD'= Transient Data trigger level
*                                   C'S ' = START command without data
*                                   C'SD' = START command with data
XTETR    DS    CL1                  Transaction Routing indicator, START
*                                   commands only. C'Y' if START issued
*                                   by Transaction Routed task.
*                                   Otherwise 'N'. 'N' for TD
XTEFS    DS    CL1                  Function Shipping indicator, START
*                                   commands only. C'Y' if START request
*                                   was Function Shipped. Otherwise 'N'.
*                                   'N' for TD requests.
XTETRAN  DS    CL4                  transaction id on request     @D2A
XTEREQTR DS    CL4                  terminal id on request        @D2A
XTECURTR DS    CL4                  For START commands, the name of the
*                                   current terminal if the command was
*                                   transaction routed, or the session
*                                   ID if the command was function
*                                   shipped.
*                                   Otherwise blanks.
XTENETIN DS    CL8                  netname of sysid, if there is    *
                                    a sysid, or blanks            @D2A
XTESYSIN DS    CL4                  sysid, if any, passed to exit    *
                                    or blanks                     @D2C
XTENETOT DS    CL8                  netname returned by exit for     *
                                    return code 4.
XTESYSOT DS    CL4                  sysid returned by the exit for   *
                                    return code 8.                @D2C
***********************************************************************
*        REGISTER DEFINITIONS                                         *
***********************************************************************
R0       EQU   0                    not used
R1       EQU   1                    base for parameter list built    *
                                    by DFHUEH
R2       EQU   2                    base for parameter list built    *
                                    by module calling the exit
R3       EQU   3                    not used
R4       EQU   4                    not used
R5       EQU   5                    not used
R6       EQU   6                    not used
R7       EQU   7                    not used
R8       EQU   8                    not used
R11      EQU   11                   base register
```

Figure 23 (Part 3 of 5). Sample program for XALTENF and XICTENF exits

```
R12      EQU   12                      TCA base
R13      EQU   13                      standard save area base
R14      EQU   14                      return address
R15      EQU   15                      entry point address
         DFHEJECT
         DFHUEXIT TYPE=EP
         DFHEJECT

DFHXTENF CSECT
         DFHVM XTENF
         ENTRY DFHXTENA
DFHXTENA DS    0H
         STM   R14,R12,12(R13)         save registers
         BALR  R11,0                   set up base register
         USING *,R11
         USING DFHUEPAR,R1             DFHUEH parameter list
*
         L     R2,UEPHMSA              calling module's register save area
         L     R2,24(R2)               calling module's register 1
*
         USING XTEPARMS,R2             calling module's parameter list
*        Could check the terminal id at this point. In this
*        program we assume it is valid. We also choose to accept
*        START requests and reject Transient Data trigger level
*        events.
*
         CLI   XTEEVENT,START          START command?
         BNE   NOTSTART                no, must be Transient Data
*
*        Accept the default netname if we are Function Shipping.
*        Otherwise build a netname.
         CLI   XTEFS,YES               Function Shipping?
         BNE   BLDNETNM                no, build a netname
*
         LH    R15,OKNETNM             accept the default netname
         B     EXIT
*
BLDNETNM DS    0H
*
*        Build a netname by taking the first character of the
*        terminal id and appending it to the characters 'CICS'.
*
         MVC   XTENETOT,=C'CICS    '
         MVC   XTENETOT+4(1),XTEREQTR
         LH    R15,OKNETNM
         B     EXIT
*
```

Figure 23 (Part 4 of 5). Sample program for XALTENF and XICTENF exits

```
NOTSTART DS    0H
         LH    R15,BAD               reject Transient Data trigger       *
                                     level events
*
EXIT     DS    0H
         L     R14,12(R13)           restore registers except 15
         LM    R0,R12,20(R13)        which contains the return code
         BR    R14
*
***********************************************************************
*        Local equates and constants
***********************************************************************
BLANKS   DC    C'    '               for sysid check
BAD      DC    H'0'                  'terminal does not exist'
OKNETNM  DC    H'4'                  'terminal exists', netname returned
OKSYSID  DC    H'8'                  'terminal exists', sysid returned
YES      EQU   C'Y'
START    EQU   C'S'
*
         DFHEND DFHXTENF
```

Figure 23 (Part 5 of 5). Sample program for XALTENF and XICTENF exits

Part 5. System enhancements

This part describes how you can enhance CICS code with user-written enhancements or variations, such as user exit routines or initialization overlays.

"Chapter 5.1. Global user exits" on page 289 describes the CICS user exit interface and the methods of incorporating user-written exit routines into CICS management programs.

"Chapter 5.2. Exit to allow modification and redirection of CICS messages" on page 327 describes an exit that allows the reformatting and redirection of CICS messages sent to transient data queues.

"Chapter 5.3. File control status exits" on page 343 describes the exits that are invoked when ENABLE, DISABLE, OPEN, and CLOSE commands are issued against a file.

"Chapter 5.4. Task-related user exits" on page 345 describes a way of communicating with non-CICS managers of recoverable system resources.

"Chapter 5.5. Writing postinitialization and termination programs" on page 369 provides guidance on writing programs to be executed during postinitialization and during system shutdown.

"Chapter 5.6. System initialization overlays" on page 373 describes the user-written overlays that may be added to the system initialization program.

"Chapter 5.7. CICS security management" on page 377 provides information on modules DFHXSP, DFHXSE, and DFHACEE, should you wish to create your own versions of them.

"Chapter 5.8. CICS monitoring facility" on page 387 describes CICS facilities for collecting information on system performance, system usage, and exceptional conditions.

"Chapter 5.9. Examining and modifying resource attributes" on page 423 describes how to provide application programs with command-level access to information about CICS resources.

"Chapter 5.10. CICS interface to JES" on page 465 describes the interface between the JES spooling capability and a CICS application.

"Chapter 5.11. Finding programs that use CICS macros" on page 481 describes a CICS-supplied program that helps you to identify application programs that use CICS macros. This is useful if you want to convert all programs to command level.

"Chapter 5.12. CEMT programming interface" on page 483 describes how to use the master terminal transaction, CEMT, from within an application program.

Chapter 5.1. Global user exits

CICS is designed to fulfil most needs of a database and data-communication system. Nevertheless, an installation can have special requirements that cannot be met by the standard system. User exits make it possible to modify the system without changing its standard user interfaces. You can customize CICS by using exit programs. These are programs that are executed at various strategic places within CICS. In these exits you can modify the subsequent execution of CICS. CICS supplies two types of exits — global user exits and task-related user exits.

The aim of exits is to isolate your own special code from the CICS supplied code, so providing enhanced function with no loss of integrity.

Global user exits

Global user exits are located within CICS management modules. You can code Assembler programs that are run at these points to modify subsequent CICS execution. In general, the exits are located at the entry and exit points of management modules, or before and after significant actions within CICS modules. "List of exits" on page 290 lists the global user exits. "Exit descriptions" on page 304 gives detailed descriptions of the exits.

The exit programs are dynamically enabled and disabled by ENABLE and DISABLE commands. See "Enabling an exit program" on page 296 and "Disabling an exit program" on page 300.

For an example of a global user exit, see "Chapter 5.2. Exit to allow modification and redirection of CICS messages" on page 327.

Most global user exit programs cannot contain CICS requests (the exceptions are programs for exits located in DFHDBP and in transaction backout programs).

Task-related user exits

You can write a CICS user exit program that acts as an interface between application programs and non-CICS recoverable system resources. Such an interface is invoked explicitly by application programs. It becomes part of the task that invoked it, and can use CICS services. This kind of exit is described in "Chapter 5.4. Task-related user exits" on page 345.

List of exits

Table 3 (Page 1 of 2). CICS exits

Exit name	Location	Description
XALTENF	*DFHALP*	When a terminal-not-known condition arises
XDBDERR	*DFHDBP*	When an error is returned from DL/I processing
XDBFERR	*DFHDBP*	When an error is returned from file control processing
XDBIN	*DFHDBP*	After each record is received
XDBINIT	*DFHDBP*	On module entry
XFCIN	*DFHFCP*	Before an input event
XFCINC	*DFHFCP*	After an input event
XFCOUT	*DFHFCP*	Before an output event
XFCOUTC	*DFHFCP*	After an output event
XFCREQ	*DFHFCP*	Before entry analysis
XFCSREQ	*DFHFCP*	Before file request analysis
XFCSREQC	*DFHFCP*	After file request analysis
XGMTEXT	*DFHGMM*	Before sending a 'Good morning' message
XICEXP	*DFHICP*	After a time interval has expired
XICREQ	*DFHICP*	Before request analysis
XICTENF	*DFHICP*	When a terminal-not-known condition arises
XISLCLQ	*DFHISP*	After a failure to establish a link to a remote system (decision for local queuing)
XJCWR	*DFHJCP*	After a journal record has been built in the buffer
XKCAWT	*DFHKCP*	After an operating system wait has ended
XKCBWT	*DFHKCP*	Before an operating system wait is to be taken
XKCDISP	*DFHKCP*	Before a task dispatch
XKCREQ	*DFHKCP*	Before request analysis
XPCABND	*DFHPCP*	Before a dump is taken
XPCFTCH	*DFHPCP*	Before control passes to an application program
XRCFCER	*DFHFCBP*	When CICS file control returns an error during recovery
XRCINIT	*DFHFCBP, DHFUSBP, DFHTCBP, DFHDLBP*	Backout program initialization and termination
XRCINPT	*DFHFCBP, DFHUSBP, DFHTCBP*	After a backout record has been read
XRCOPER	*DFHFCBP*	After an error in opening a file during recovery
XSCREQ	*DFHSCP*	Before request analysis
XTCATT	*DFHTCP*	Before a task is attached
XTCIN	*DFHTCP*	After an input event

Table 3 (Page 2 of 2). CICS exits		
Exit name	**Location**	**Description**
XTCOUT	*DFHTCP*	Before an output event
XTCRDAT	*DFHTCP*	After a 2741 read attention has occurred
XTCTIN	*DFHTCP*	After a TCAM input event
XTCTOUT	*DFHTCP*	Before a TCAM output event
XTDCOUT	*DFHTDP*	Before a write to a CICS system queue
XTDIN	*DFHTDP*	Before an input event
XTDOUT	*DFHTDP*	Before an output event
XTDREQ	*DFHTDP*	Before request analysis
XTSIN	*DFHTSP*	After an input event
XTSOUT	*DFHTSP*	Before an output event
XTSREQ	*DFHTSP*	Before request analysis
XXRSTAT	*DFHXRA*	Before status information is written to the CAVM dataset
XZCATT	*DFHZCP*	Before a task is attached
XZCIN	*DFHZCB*	After a VTAM input event
XZCOUT	*DFHZCB*	Before a VTAM output event
XZCOUT1	*DFHZCB*	Before a VTAM message is broken up into RUs

User exit interface

The user exit interface has the following characteristics:

- The use of global user exits requires that the CICS system is initialized with the SIT operands EXEC=YES and EXITS=YES, for command-level support and user-exit support, respectively.

  For the transaction backout exits, you must also specify the DFHSIT operand (or override) TBEXITS=(n1,n2,n3,n4), in which n1,n2,n3, and n4 are the names of your user programs for XRCINIT, XRCINPT, XRCFCER, and XRCOPER.

  To enable the terminal-not-found exit XALTENF during CICS initialization, you must specify the DFHSIT operand (or override) ALEXIT. See the *CICS/MVS Operations Guide*.

- The exit program is compiled and linked as if it were a normal macro-level Assembler program - but it cannot contain any calls to CICS services. The program must reside below the 16MB line, and must be defined in the CSD or the PPT. If you attempt to enable an exit program that resides above the 16MB line, the INVEXITREQ exceptional condition is raised, as for any error in the ENABLE command (see "Error responses" on page 302).

- Two or more exit programs can be invoked at a single exit (see "Use of multiple programs at one exit" on page 293), or, conversely, a single exit program can be associated with two or more exits.

- Some of the exit programs can provide a return code. For details of exit processing and the parameters passed by CICS, see "Exit descriptions" on page 304.
- The use of an EXIT is controlled by EXEC CICS ENABLE and EXEC CICS DISABLE commands, which can be placed in any type of command-level program, or run using the CECI transaction.

  When enabling an exit program, you can ask CICS to provide a global work area for use by the exit program. An exit program can have its own global work area, or two or more exit programs can share a work area. It is recommended that the global work area is shared only by the user exits associated with the same management module. The address and length of the work area are passed as parameters to the exit programs that use it. A work area is freed when all exit programs that use it are disabled. The way you do this is described in "Enabling an exit program" on page 296 and "Disabling an exit program" on page 300.
- If the transaction issues EXEC CICS ENABLE or DISABLE commands and then abends, any change of exit status is not backed out. Similarly, the status of an exit is not keypointed. Therefore, exit status is not maintained over any type of CICS restart.
- When the CICS system is being used in several regions, exit activity is independent for each region even if the management module is shared. Therefore, exit programs must be enabled in each region in which they are to be used.

User exit handler

If there is an exit program associated with an exit point, the code at the exit point invokes the user exit handler (DFHUEH). DFHUEH saves the calling module's registers in part of the task's own storage. It then checks if the exit program is ready to be started. If it is not ready, DFHUEH returns control at once. If the exit program is ready, this must also mean that it is already resident in virtual storage and its entry address is known. (If the program is not defined as resident, it will have been loaded and made "temporarily resident" by the exit ENABLE process.) See "Enabling an exit program" on page 296. Before invoking the program, DFHUEH builds a standard parameter list DFHUEPAR (see page 295), which is passed to the exit via R1. (The management module registers are addressed by UEPHMSA in this parameter list).

If global user exit tracing is active, 'before exit' trace records are written before invoking the exit program and 'after trace' records are written when control returns from to the exit program.

This process is then repeated for any other active routines for the exit (see "Use of multiple programs at one exit" on page 293). When all the exit programs have been run, DFHUEH restores the registers and returns to the exit point in the CICS management module.

Use of multiple programs at one exit

You may want to use more than one program at an exit. You can have two or more application packages that both supply programs to utilize the same CICS exit. Although such programs can function independently (for example, they can be activated and started without reference to each other), you should note the following about activation and the use of return codes.

Although each program will only be called at an exit if it has been enabled, the order of invocation, once multiple programs have been started, is the order of activation. Full details about ENABLE are given in "Enabling an exit program" on page 296. Where programs work on the same data area, the precise order of invocation must be considered. For example, in a terminal control output exit, multiple programs might be manipulating the same message in different ways, depending upon the way an earlier exit program acted.

Return code management is more complicated than it is for single programs. For single programs, a return code other than zero is passed back in register 15, provided that the value is a positive multiple of 4 within the range defined by the exit point. Any program subsequently called at the same exit can access the "current return code" provided by the last program. The new program's user parameter list includes the address of the "current" value in field UEPCRCA.

The following rules apply to return codes if a second user exit program sets a different return code from that selected by the previous program.

- If the new program sets the same return code in register 15 as the currently-established return code (UEPCRCA-addressed), then CICS acts on that value.
- If the new program sets a value in register 15 different from the current value in UEPCRCA, CICS ignores both values and resets the "current return code" to the default (zero).
- If the new program sets an eligible value in register 15 and changes the "current value" field to match (as addressed by UEPCRCA), then the new value will be adopted and passed on to the next program (if any) or back to the CICS calling module.

Using an exit

Global user exit programs are reentrant Assembler language programs residing below the 16MB line. In general, these programs are effectively part of the CICS nucleus code, and they cannot use CICS services. Therefore, you code and link your exit as if it were a non-CICS module (that is, you treat it as a macro-level Assembler program which does not issue any CICS calls). However, exits in the backout programs are effectively within application code, so macro-level services are available.

You can issue MVS system requests within the exit, but beware that ill-considered use of MVS function may seriously effect the efficient running of CICS.

Information is passed to the exit by means of registers, and a return code may be provided to affect subsequent CICS operation. You can use multiple exit programs at the same exit point; if you do, you must know how return codes are handled in this case (see "Use of multiple programs at one exit" on page 293).

An exit program is not used by CICS until it is activated by an EXEC ENABLE command. The re-entrant requirement is met by the provision of a work area, the length of which is specified when the exit is enabled. Serious errors are likely to occur if the work area length is incorrectly specified. The obtained work areas are maintained across exit invocations.

Global user exits are release-dependent. No guarantee is given that either the exit point itself, or its function will be unchanged between releases of CICS. You should ensure that you always check the exit documentation and regenerate all your exits when migrating to a new release of CICS.

Exit program conventions

Terminology note

In this section, and for the rest of the chapter, the notation R***n*** is used to represent register ***n***. For example, R5 represents register 5.

This section describes the programming interface for the global user exits. Your exit program must conform to the standards in this section.

1. Upon entry to the exit routine, the following registers have defined values :

Register	**Value**
R1	The address of the exit parameter list (mapped by DFHUEXIT TYPE=EP)
R12	The address of the user TCA of the invoking task
R13	The address of a standard MVS save area. It does ***not*** point to the CSA
R14	The return address for your exit routine
R15	The entry address of the exit routine.

   The other registers are undefined. However, the individual exits may provide additional information in these registers.

2. The first thing the exit routine must do is save the registers in the save area addressed by R13.

3. R1 points to the exit parameter area which is mapped by the DFHUEXIT TYPE=EP macro. You should not alter this parameter list.

```
DFHUEPAR DSECT
UEPEXN   DS    A      ADDRESS OF EXIT NUMBER
UEPGAA   DS    A      ADDRESS OF GLOBAL AREA
*                      (ZERO = NO WORK AREA)
UEPGAL   DS    A      ADDRESS OF GLOBAL AREA LENGTH
UEPCRCA  DS    A      ADDRESS OF CURRENT RETURN-CODE
UEPTCA   DS    A      ADDRESS OF TCA
UEPCSA   DS    A      ADDRESS OF CSA
UEPEPSA  DS    A      ADDRESS OF REGISTER SAVE AREA
*                      FOR USE BY EXIT PROGRAM
UEPHMSA  DS    A      ADDRESS OF SAVE AREA USED FOR
*                      HOST MODULE'S REGISTERS
```

UEPEXN points to a XL1 field that identifies the global user exit which has been taken. Although user names should not change across releases, identification numbers do. Consequently, you must not assume that a named exit will always have the same identification number. The DFHUEXIT TYPE=EP macro call defines the identification numbers which you must use for this purpose.

UEPGAA and UEPGAL
point to the global work area and its halfword length, respectively. These are set when the exit is enabled. You must ensure that the length of the workarea is sufficient or serious errors will occur. The workarea can be shared between all the exit routines within an exit program, or each routine can have its own copy. See "Enabling an exit program" on page 296 for details.

UEPCRCA points to a halfword that contains the return code from the exit program. This is set to zero for the first exit program that is invoked at a user exit. If there is more than one program called at a user exit, this field contains the current return code for reference by the next program. See also "Use of multiple programs at one exit" on page 293.

UEPTCA and UEPCSA
contain the address of the current user TCA and the CSA, respectively. Remember that R13 in the exit does ***NOT*** point to the CSA. The values of R12/13 within UEPHMSA also point to the TCA & CSA.

UEPEPSA points to a save area, for the user's exit program to use, to preserve registers at entry. When the exit program is entered, register 13 is also pointing to this area. The convention is to save registers 14, 15, 0 ...12 at offset 12 (decimal) onward.

UEPHMSA points to the "host module save area" that DFHUEH first used when called by the CICS management module. Values for registers 14, 15, 0 ...13 are stored in this order from offset 12 (decimal) in this area.

Apart from register 15, which contains the return code, the values in this save area are used to reload the registers when returning from DFHUEH to the calling CICS module. They should not be corrupted.

4. If you wish your exit routine to issue MVS service calls, you must set R13 to point to a new MVS-format savearea, call the service, and reset R13 on return. The suggested location of this savearea is within the exit's workarea. Remember that use of MVS services can have an considerable impact on the efficiency of CICS processing.
5. To end the exit routines processing, you must restore all registers apart from R15 from the R13 save area. If the exit permits the setting of a return code, this should be placed in R15. Then branch to R14 to return to CICS.
6. If the exit permits the setting of a return code, R15 is checked for validity—if it is invalid, a default is taken. UEPCRCA is provided for use when more than one program is invoked at the exit. See "Use of multiple programs at one exit" on page 293 for details.

Enabling and disabling an exit program

You can dynamically control exit activity by ENABLE and DISABLE commands in a CICS command level program written in COBOL, PL/I or assembler language. The mechanism for enabling and disabling exits has a number of optional facilities that make the commands flexible enough to meet the needs of different application packages.

The commands listed in this chapter are intended to be used only by the system programmer, and not by the application programmer.

The general rules about the use of commands in CICS application programs are given in the *CICS/MVS Application Programmer's Reference* manual.

Note: The enabling and disabling of an exit program has no effect on the enable or disable status for the program in the CSD file or in the PPT entry.

Enabling an exit program

An exit program is enabled in three stages, as follows:

- Establish — to load the exit program or specify its entry address, and obtain a global work area for use by the exit program, if requested.
- Activate — to associate the exit program with an exit.
- Start — to make the exit program available for execution.

These three stages can be performed in a single ENABLE command, or can be split across two or more commands. You can specify only one exit in a single ENABLE command. Therefore, if an exit program is to be activated for two or more exits, delayed use of the START option allows execution of the exit program to be suppressed until all the ENABLE commands are completed. Examples are given after the syntax and options.

ENABLE command

```
EXEC CICS ENABLE
     PROGRAM(name)
     [EXIT(name)]
     [START]
     [ENTRYNAME(name)]
     [ENTRY(pointer-value)]
     [GALENGTH(data-value) | GAENTRYNAME(name)]
```

EXEC CICS ENABLE
Specifies that all or part of the establish-activate-start sequence is to be performed for an exit program.

PROGRAM(name)
Specifies the name of the load module of the exit program. You must use this option with every ENABLE command. The first time you specify PROGRAM, CICS will enable the code associated with the program name. The name can be any character string up to eight bytes, and it must be the name of a program in the CSD file or in the PPT.

EXIT(name)
Specifies the exitid of the exit for which the exit program is to be activated (see "List of exits" on page 290). CICS does not check that the management program containing this exit is present in the CICS system. If the management program is not present, the ENABLE command can complete normally but the exit can never be used.

START
Specifies that the exit program is to be made available for execution. Where several exits are associated with an exit program, omission of this operand allows exit program execution to be suppressed until sufficient exits have been associated with the exit program for it to execute correctly. After you have STARTed the code for an exit or number of exits, you can continue to activate the code for further exits.

ENTRYNAME(name)
Specifies the name of this entry to the exit program. So you can have a logical name for a piece of user exit code that is different from the physical load module name. This name need not be defined in the CSD or PPT. It must be unique among the enabled entry names. If omitted, the name will be taken from the mandatory PROGRAM operand. Its presence ***does not*** require the ENTRY keyword to be specified. The same combination of ENTRYNAME/PROGRAM that is specified on the initial ENABLE must be used on subsequent ENABLE, DISABLE, and EXTRACT EXIT commands directed to the named entry.

ENTRY(pointer-value)
Specifies the entry address of the exit program. If this operand is specified, CICS assumes that the exit program is already loaded and will not attempt to load it, nor will it attempt to delete it when the exit program is disabled. ENTRY is only valid for the first ENABLE of a program. You must still specify

PROGRAM, and the program must be defined in the CSD or the PPT. The specified address must be within the virtual storage range occupied by the exit program. When using this option, the program must be permanently resident, or must be already loaded and remain resident while it is being used in an exit. If this operand is not specified, the exit program is loaded by CICS, the entry address returned from the load is used, and CICS will delete the exit program when it is disabled.

GALENGTH(data-value)
Specifies the length, in bytes, of the global work area that is to be provided by CICS for this exit program. If a data variable is specified, it must represent a halfword binary data item. Valid lengths are 1 through 32767. The work area will be initialized to binary zeros.

GAENTRYNAME(name)
Specifies the name of a currently-enabled entry whose global work area is also to be used by the entry being enabled. The name is normally a program name, but, if ENTRYNAME was specified for the code that is now going to share its global work area, you use the entryname instead. The entry specified must own the work area (that is, GALENGTH must have been specified when the entry was enabled). If a work area is shared by two or more entries, it is not released until all these entries are disabled. However, after the owning entry is disabled, no new entry can share the work area.

Note that the use of GAENTRYNAME makes disabling exits more complex. For example, even if the first exit code is DISABLEd with EXITALL, CICS has to keep its global area as long as a second piece of code needs it. In such situations it may be good practice to code STOP even on a DISABLE with EXITALL, as a precaution against restarting.

GALENGTH and GAENTRYNAME are mutually exclusive. If both operands are omitted, no work area is provided.

Note: On the second and subsequent ENABLE commands for a particular exit program, ENTRY, GAENTRYNAME, and GALENGTH must not be specified and either EXIT or START, or both, must be specified.

Examples: The establish-activate-start sequence for an exit program can be done in a single ENABLE command, or in two or more ENABLE commands, as follows:

1. In one command:

   ```
   EXEC CICS ENABLE PROGRAM('EP1') EXIT('XFCOUT') START
   ```

   The above command loads exit program EP1, activates it for exit XFCOUT, and starts the exit program. No work area is obtained.

   ```
   EXEC CICS ENABLE PROGRAM('EP2') EXIT('XKCDISP')
   START ENTRY(EADDR) GALENGTH(500)
   ```

   The above command assumes that exit program EP2 is already loaded, with its entry address in EADDR. It activates EP2 for exit XKCDISP and starts the exit program. A work area of 500 bytes is obtained.

2. In two or more commands:

   ```
   EXEC CICS ENABLE PROGRAM('EP3') EXIT('XTDOUT')
   GAENTRYNAME('EP2')

   EXEC CICS ENABLE PROGRAM('EP3') EXIT('XTDIN')

   EXEC CICS ENABLE PROGRAM('EP3') EXIT('XTDREQ') START
   ```

   The first command above loads exit program EP3 and activates it for exit XTDOUT; EP3 will be able to use the work area obtained for EP2. The second command activates EP3 for exit XTDIN. The third command activates EP3 for exit XTDREQ and starts the exit program.

3. The above examples are straightforward demonstrations of the use of the ENABLE command. This example shows you how to have one load module which contains a number of sections of code, each operating at various exit points, and each of which can have its own global work area.

   You can "enable" the same load module more than once, if you use a different ENTRYNAME each time. You thus associate a different section of user code with each ENTRYNAME, enable the module with appropriate ENTRY point and GALENGTH, and activate it, under the right ENTRYNAME, for those exits appropriate to the code.

   Module EP4 contains two separate sections of code, one to be used at the XFCIN exit, and one for XFCOUT. The entry point addresses for the two sections are held in two successive fullwords, either just before, or just after, the "entry point" of the module EP4. EP4 is defined in the CSD or PPT. For simplicity in the example, EP4 is permanently resident in virtual storage.

   At run time, the enabling program brings up the exits in the following way, finding the module entry point first:

   ```
   EXEC CICS LOAD PROGRAM('EP4') ENTRY(modaddr)
   ```

   where modaddr is a suitable work area. Then it locates the address constants for the two code sections, and copies them into two work fields, INADDR and OUTADDR. Then the two exits are brought up:

   ```
   EXEC CICS ENABLE PROGRAM('EP4') GALENGTH(length1)
   ENTRYNAME('FCIN') ENTRY(INADDR) EXIT('XFCIN') START

   and

   EXEC CICS ENABLE PROGRAM('EP4') GALENGTH(length2)
   ENTRYNAME('FCOUT') ENTRY(OUTADDR) EXIT('XFCOUT') START
   ```

   Note that there are two independent "enables" of EP4, each associated with a different exit. So the three processes of ENABLE (enable, activate, start) are combined, for each exit.

   If you do not want separate global work areas, then coding GAENTRYNAME('FCIN') instead of GALENGTH on the second ENABLE makes the two sections share one area.

Disabling an exit program

An exit program is disabled in three stages, as follows:

- Stop – to make the exit program unavailable for execution. It can no longer be invoked by an exit point.
- Deactivate – to dissociate the exit program from an exit, one exit at a time, or all at once.
- Release – to delete the exit program (if it had been loaded to serve as an exit) and release a global work area, if appropriate.

As for ENABLE, these three stages can be performed in a single DISABLE command, or can be split across two or more commands.

DISABLE command

```
EXEC CICS DISABLE
     PROGRAM(name)
     [ENTRYNAME(name)]
     [EXIT(name) | EXITALL]
     [STOP]
```

EXEC CICS DISABLE
Specifies that all or part of the stop-deactivate-release sequence is to be performed for an exit program.

PROGRAM(name)
Specifies the name of the exit program. You must code this option with every DISABLE command.

ENTRYNAME(name)
Specifies an enabled entry name. Its default is taken from the PROGRAM operand. For successful execution of the DISABLE command, it is necessary that the same combination of ENTRYNAME/PROGRAM be specified as was specified in the original ENABLE.

EXIT(name)
Specifies the exitid of an exit for which the exit program is to be deactivated (see "Exit descriptions" on page 304). The exit program will not be disabled.

EXITALL
Specifies that the exit program is to be deactivated for all exits for which it is active. The exit program will then be disabled. EXITALL implies STOP. After using EXITALL, you are back in the situation that existed before you enabled the exit, unless you are sharing global work areas using the GAENTRYNAME option. This situation is described under the GAENTRYNAME option of ENABLE.

STOP

Specifies that the exit program is to be stopped before any deactivations are done. Where several exits are associated with an exit program, this operand allows exit program execution to be suppressed while sufficient exits are associated with the exit program for it to execute correctly. You can reverse the effect of DISABLE STOP by an ENABLE START command.

At least one of the operands EXIT, EXITALL, and STOP must be specified. EXIT and EXITALL are mutually exclusive; if both are omitted, no deactivations are performed.

Examples: The stop-deactivate-release sequence is usually performed in a single DISABLE command. However, it is possible to use the command to stop or deactivate an exit program without disabling it. Here are some examples:

1.
   ```
   EXEC CICS DISABLE PROGRAM('EP2') STOP
   ```
   This command simply stops exit program EP2. It can be restarted later by issuing ENABLE PROGRAM('EP2') START.

2.
   ```
   EXEC CICS DISABLE PROGRAM('EP3') EXIT('XTDREQ')
   ```
   This command deactivates exit program EP3 for exit XTDREQ. Any other exits associated with EP3 will be unaffected. Subsequently, to fully disable EP3, you must issue:

3.
   ```
   EXEC CICS DISABLE PROGRAM('EP3') EXITALL
   ```

This command stops exit program EP3, deactivates all its exits, and then deletes the exit program.

Locating the exit work area

The address of the workarea is passed to the exit in UEPGAA. No distinction is made between an area unique to the exit and an area shared between the exit and a program.

Command-level application programs can obtain the address of this area. If an application program updates the work area, it must ensure that an exit is not currently using the area.

Accessing a work area

Application programs can obtain the address and length of the global work area that is owned or shared by a specific exit program by means of the EXTRACT EXIT command.

EXTRACT EXIT command

```
EXEC CICS EXTRACT
     EXIT
     PROGRAM(name)
     [ENTRYNAME(name)]
     GASET(pointer-ref)
     GALENGTH(data-area)
```

EXEC CICS EXTRACT
Specifies that information is to be extracted from a CICS control block.

EXIT
Specifies the type of control block.

PROGRAM(name)
Specifies the name of an exit program. The address and length of this exit program's global work area is to be extracted from the control block. The exit program can either own or share the work area.

ENTRYNAME(name)
Specifies the name of an enabled entry. By default it takes the value specified by the PROGRAM operand. The same combination of ENTRYNAME/PROGRAM must be specified as on the original ENABLE command.

GASET(pointer-ref)
Specifies the variable that is to be set to the address of the global work area used by the exit program.

GALENGTH(data-area)
Specifies the variable that is to be set to the length of the global work area used by the exit program. It must be a halfword binary data item.

Example:

```
EXEC CICS EXTRACT EXIT PROGRAM('EP2') GASET(WORKA) GALENGTH(WORKL)
```

This command puts the address of the work area used by exit program EP2 in the pointer referenced by WORKA, and puts the length of the work area in the data item referenced by WORKL.

Error responses

Check the EIBRCODE

If an INVEXITREQ condition occurs, check EIBRCODE. The source of the error may be outside the command; for example, an EIBRESP of X'8000' is generated if the exit program is defined to reside above the 16MB line

For details of how to code tests for exceptional conditions using DFHRESP, see the *CICS/MVS Application Programmer's Reference* manual.

All errors generated by the EXEC CICS ENABLE, EXEC CICS DISABLE, and EXEC CICS EXTRACT EXIT commands are translated into the INVEXITREQ condition. The default action for INVEXITREQ is to terminate the transaction with abend code AEY0.

If you are handling INVEXITREQ, or using RESP processing, EIBRCODE byte 0 will be X'80' if an error occurred. EIBRCODE bytes 1-2 contain the error cause as follows :

EIBRCODE bytes 1-2	Command	Meaning
X'8000'	ENABLE	A CICS program management request implicit in the ENABLE command has raised the PGMIDERR condition. This means that the program specified in PROGRAM is not in the PPT, or is not in the load library, or its PPT entry has been disabled. PGMIDERR also occurs in an MVS/XA system if an application program executing in 24-bit mode issues an ENABLE command for a program residing at an address above the 16MB line.
X'4000'	ENABLE	Exitid is invalid.
	DISABLE	Exitid is invalid.
X'2000'	ENABLE	Program specified in ENTRYNAME (or defaulted to PROGRAM argument) is already enabled.
X'1000'	ENABLE	Program specified in ENTRYNAME (or defaulted to PROGRAM argument) is already active for the exitid specified in EXIT.
X'0800'	ENABLE	Program specified in GAENTRYNAME is not enabled.
X'0400'	ENABLE	Program specified in GAENTRYNAME does not own a work area.
	EXTRACT EXIT	Program has no work area.
X'0200'	DISABLE	Program is not enabled.
	EXTRACT EXIT	Program is not enabled.
X'0100'	DISABLE	Program has not been activated for exitid specified in EXIT.
X'0080'	DISABLE	Program is currently invoked by another task (see note).
X'0040'	All three	User exit interface was not initialized.

Note: The INVEXITREQ condition with X'0080' in bytes 1 and 2 can occur only if a task switch has occurred in the exit program due to a request for a CICS service. The normal action for this condition is to retry the DISABLE request. However, if such an exit program terminates abnormally, its use count will remain greater than zero and it cannot be disabled or deactivated, but it can be stopped.

Exit descriptions

The following is a list of the global user exits. (Additional exits are available to users of the data tables feature. These are described in the *CICS/MVS Data Tables Guide*).

XALTENF

Location	*DFHALP*
Description	When a "terminal-not-known" condition arises

Exit specific registers
None

Valid return codes

Value	Action
0	Terminal does not exist
4	NETNAME returned
8	SYSID returned

Processing information:

1. This exit is fully described in "Chapter 4.9. Exits for "terminal-not-known" condition" on page 277.
2. UEPHMSA's R1 points to an exit specific parameter list.

XDBDERR

Location	*DFHDBP*
Description	When an error is returned from DL/1 processing

Exit specific registers

Register	**Addresses**
R3	Dynamic log record

Valid return codes

Value	**Action**
0	Suppress further DL/1 backout
4	Suppress further DL/1 backout

Processing information:

1. This exit is fully described in "Chapter 2.4. Writing dynamic transaction backout exits" on page 53.
2. UEPHMSA's R1 points to a FCT entry (if any).

XDBFERR

Location	*DFHDBP*
Description	When an error is returned from File Control processing

Exit specific registers

Register	**Addresses**
R3	Dynamic log record
R6	The FWA (if any)

Valid return codes

Value	**Action**
0	Accept the error and continue
4	Ignore the error and continue
8	Retry

Processing information:

This exit is fully described in "Chapter 2.4. Writing dynamic transaction backout exits" on page 53.

XDBIN

Location	*DFHDBP*
Description	After each record is received

Exit specific registers

Register	**Addresses**
R3	Dynamic log record (except for DL/1)

Valid return codes

Value	**Action**
0	Continue processing
4	Ignore the record

Processing information:

This exit is fully described in "Chapter 2.4. Writing dynamic transaction backout exits" on page 53.

XDBINIT

Location	*DFHDBP*
Description	On module entry

Exit specific registers
None

Valid return codes

Value	**Action**
0	Continue with backout
4	Suppress DL/1 backout
8	Suppress ALL backout

Processing information:

This exit is fully described in "Chapter 2.4. Writing dynamic transaction backout exits" on page 53.

XFCIN

Location *DFHFCP*
Description Before an input event

Exit specific registers

Register	**Addresses**
R9	FCT entry

Valid return codes
None

XFCINC

Location *DFHFCP*
Description After an input event

Exit specific registers

Register	**Addresses**
R9	FCT entry
R11	VSWA or the FIOA

Valid return codes
None

XFCOUT

Location *DFHFCP*
Description Before an output event

Exit specific registers

Register	**Addresses**
R9	FCT entry
R10	VSWA or FIOA

Valid return codes
None

XFCOUTC

Location	*DFHFCP*
Description	After an output event

Exit specific registers

Register	**Addresses**
R9	FCT entry
R11	VSWA or FIOA

Valid return codes
None

XFCREQ

Location	*DFHFCP*
Description	Before entry analysis

Exit specific registers
None

Valid return codes
None

Processing information:

The entry information is in the FC overlay in the TCA.

XFCSREQ

Location *DFHFCS*
Description Before file-request analysis

Exit specific registers

Register	Addresses
R6	Request byte
R11	FCT entry

Valid return codes

Value	Action
0	Continue with Request
4	Suppress the Request

Processing information:

See "Chapter 5.3. File control status exits" on page 343 for details of this exit.

XFCSREQC

Location *DFHFCS*
Description After file-request analysis

Exit specific registers

Register	Addresses
R6	Request byte
R7	Response byte
R11	FCT entry

Valid return codes
None

Processing information:

See "Chapter 5.3. File control status exits" on page 343 for details of this exit.

XGMTEXT

Location	*DFHGMM*
Description	Before sending a "Good Morning" message

Exit specific registers

Register	**Addresses**
R3	TIOA
R11	TCTTE

Valid return codes
None

XICEXP

Location	*DFHICP*
Description	After a time interval has expired

Exit specific registers

Register	**Addresses**
R8	ICE that has just expired

Valid return codes
None

XICREQ

Location *DFHICP*
Description Before request analysis

Exit specific registers
None

Valid return codes
None

Processing information:

The entry information is in the IC overlay in the TCA.

XICTENF

Location *DFHICP*
Description When a "terminal-not-known" condition arises

Exit specific registers
None

Valid return codes

Value	**Action**
0	Terminal does not exist
4	NETNAME returned
8	SYSID returned

Processing information:

1. This exit is fully described in "Chapter 4.9. Exits for "terminal-not-known" condition" on page 277.
2. UEPHMSA's R1 points to an exit-specific parameter list.

XISLCLQ

Location *DFHISP*
Description After a failure to establish a link to a remote system fails (decision for local queuing)

Exit-specific registers
None

Valid return codes

Value	**Action**
0	Act according to the PCT LOCALQ setting
4	Locally queue this request
8	Return SYSIDER to the initiator of the request

Processing information

UEPHMSA's R1 addresses this parameter list:

Offset (Hex)	**Is an AL4 which addresses**
0	TCTSE for the link
4	PCT entry for the transaction (may be 0)
8	Request parameter list (may be above the 16MB line)

XJCWR

Location *DFHJCP*
Description After a journal record has been built in the buffer

Exit-specific registers

Register	**Addresses**
R7	Journal record just built
R11	JCT entry

Valid return codes
None

XKCAWT

Location *DFHKCP*
Description After an MVS wait has ended

Exit-specific registers

Register	**Addresses**
R6	Return code from the MVS SYSEVENT macro for the MVS WAIT issued before the exit was taken

Valid return codes

Value	**Action**
0	Continue
8	Issue a MVS SYSEVENT macro to suppress address-space swapping

Processing information:

RC(8) may interfere with CICS processing. Issue it with care.

XKCBWT

Location *DFHKCP*
Description Before an MVS wait is to be taken

Exit-specific registers
None

Valid return codes

Value	Action
0	Continue
4	Issue a MVS SYSEVENT macro to allow address-space swapping

Processing information:

RC(4) may interfere with CICS processing. Issue it with care.

XKCDISP

Location *DFHKCP*
Description Before a task dispatch

Exit-specific registers

Register	Addresses
R3	DCA to be dispatched

Valid return codes
None

XKCREQ

Location *DFHKCP*
Description Before request analysis

Exit-specific registers
None

Valid return codes
None

Processing information:

The entry information is in the KC overlay in the TCA.

XPCABND

Location	*DFHPCP*
Description	Before a dump is taken

Exit-specific registers

None

Valid return codes

Value	**Action**
0	Take the dump
4	Suppress the dump

Processing information:

The entry information is in the DC overlay in the TCA.

XPCFTCH

Location	*DFHPCP*
Description	Before control passes to an application program

Exit-specific registers

Register	**Addresses**
R8	PPT entry for the program

XRCFCER

Location	*DFHFCBP*
Description	When File Control returns an error during recovery

Exit-specific registers

Register	**Addresses**
R2	Not used for addressing, but contains an error code — see "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59
R7	FBO entry
R9	FWA (if any)
R10	A copy of the log record
R11	FCT entry (where applicable)

Valid return codes
None

Processing information:

This exit is fully described in "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.

XRCINIT

Location	*DFHDLBP, DFHFCBP, DFHTCBP, DFHUSBP*
Description	Backout program initialization and termination

Exit-specific registers

Register	**Addresses**
R2	Not used for addressing, but contains an invocation code — see "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.
R7	Backout Table (if any)

Valid return codes
None

Processing information:

This exit is fully described in "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.

XRCINPT

Location *DFHFCBP, DFHTCBP, DFHUSBP*
Description After a backout record has been read

Exit-specific registers

Register	**Addresses**
R7	Backout table entry (if any)
R10	Current log record

Valid return codes

Value	**Action**
0	Take the default action—see "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59
4	Ignore the log record

Processing information:

This exit is fully described in "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.

XRCOPER

Location *DFHFCBP*
Description After an error in opening a File during recovery

Exit-specific registers

Register	**Addresses**
R7	FBO entry

Valid return codes
None

Processing information:

1. This exit is fully described in "Chapter 2.6. User-written exits for resource backout or recovery at emergency restart" on page 59.
2. This exit is only given control if the operator replies "GO" to message DFH5708.
3. After return from this exit, the FBO entry is marked "No Action."

XSCREQ

Location *DFHSCP*
Description Before request analysis

Exit-specific registers
None

Valid return codes
None

Processing information:

1. The entry information is in the SC part of the TCA
2. This exit is not taken for LIFO storage requests.

XTCATT

Location *DFHTCP*
Description Before a task is attached

Exit-specific registers

Register	**Addresses**
R2	TCTTE
R4	TIOA

Valid return codes
None

Processing information:

1. UEPHMSA's R1 points to the TCTLE (as does TCTTELEA).
2. This exit is also driven by TCAM, see "Chapter 4.2. The CICS/TCAM interface" on page 205.

XTCIN

Location *DFHTCP*
Description After an input event

Exit-specific registers

Register	**Addresses**
R2	TCTTE
R4	TIOA

Valid return codes
None

Processing information:

UEPHMSA's R1 points to the TCTLE (as does TCTTELEA).

XTCOUT

Location *DFHTCP*
Description Before an output event

Exit-specific registers

Register	**Addresses**
R2	TCTTE
R4	TIOA

Valid return codes
None

Processing information:

UEPHMSA's R1 points to the TCTLE (as does TCTTELEA).

XTCRDAT

Location *DFHTCP*
Description After a 2741 Read Attention has occurred

Exit-specific registers

Register	**Addresses**
R2	TCTTE
R4	TIOA

Valid return codes
None

XTCTIN

Location *DFHTCP*
Description After a TCAM input event

Exit-specific registers

Register	**Addresses**
R2	TCTTE
R4	TIOA

Valid return codes

Value	**Action**
0	CICS will format the TCAM header
4	CICS will not format the TCAM header

Processing information:

1. UEPHMSA's R1 points to the TCTLE (as does TCTTELEA).
2. See "Chapter 4.2. The CICS/TCAM interface" on page 205 for operation of this exit.

XTCTOUT

Location *DFHTCP*
Description Before a TCAM output event

Exit-specific registers

Register	Addresses
R2	TCTTE
R4	TIOA

Valid return codes

Value	Action
0	CICS will format the TCAM header
4	CICS will not format the TCAM header

Processing information:

1. UEPHMSA's R1 points to the TCTLE (as does TCTTELEA).
2. See "Chapter 4.2. The CICS/TCAM interface" on page 205 for operation of this exit.

XTDCOUT

Location *DFHTDP*
Description Before a write to a CICS system queue

Exit-specific registers
None

Valid return codes

Value	Action
0	Write the Message to the specified queue (XTDOUT is invoked if required)
4	Do not write the message

Processing information:

1. UEPHMSA's R1 addresses this parameter list:

Offset (Hex)	Contains
0	Queue name (CL4)
4	Address of data to be written (AL4)
8	Length of data to be written (F)

2. See "Chapter 5.2. Exit to allow modification and redirection of CICS messages" on page 327 for details of this exit.

XTDIN

Location	*DFHTDP*
Description	After an input event

Exit-specific registers

Register	**Addresses**
R3	DCT entry

Valid return codes
None

Processing information:

This exit is not taken for extrapartition queues.

XTDOUT

Location	*DFHTDP*
Description	Before an output event

Exit-specific registers

Register	**Addresses**
R3	DCT entry

Valid return codes
None

Processing information:

This exit is not taken for extrapartition queues.

XTDREQ

Location *DFHTDP*
Description Before request analysis

Exit-specific registers

Register	**Addresses**
R3	DCT entry

Valid return codes
None

Processing information:

The entry information is in the TD overlay in the TCA.

XTSIN

Location *DFHTSP*
Description After an input event

Exit-specific registers

Register	**Addresses**
R4	TSIOA (after the SAA)
R8	TSGID

Valid return codes
None

XTSOUT

Location *DFHTSP*
Description Before an output event

Exit-specific registers

Register	**Addresses**
R4	TSIOA (after the SAA)
R8	First TSGID

Valid return codes
None

XTSREQ

Location *DFHTSP*
Description Before request analysis

Exit-specific registers

Register	Addresses
R4	TSIOA (after the SAA) (if present)

Valid return codes
None

Processing information:

The entry information is in the TS overlay in the TCA.

XXRSTAT

Location *DFHXRA*
Description Before status information is written to the CAVM dataset

Exit-specific registers
None

Valid return codes

Value	Action
0	Update the status information
4	Do not update the status information
8	Display message DFH0606, and abend CICS U0208 without a dump
12	Display message DFH0606, and abend CICS U0208 with a dump

Processing information:

1. UEPHMSA's R1 addresses this parameter list:

Offset (Hex)	Contains
0	Generic APPLID (CL8)
8	Specific APPLID (CL8)
10	VTAM domain ID (CL4)
14	Error ID (CL4).

2. If VTAM has failed, the VTAM domain ID will be 'ZCbb' and the error Id will be either '2307' (failure notified via RPL Completion) or '3443' (failure notified via TPEND).
3. See *CICS/MVS XRF Guide* for details of this exit.

XZCATT

Location	*DFHZCP*
Description	Before a task is Attached

Exit-specific registers

Register	**Addresses**	
R2	Following parameter list:	
	Offset (Hex)	**Contains**
	0	Address of the LU6.1 DPN or LU6.2 TPN (may be 0) (AL4)
	4	Length of the LU6.1 DPN or LU6.2 TPN (may be 0) (XL7)
	5	TRANSID about to be attached (CL4)
R8	TIOA (may be 0)	
R10	TCTTE	

Valid return codes
None

Processing information:

1. You may change R2's TRANSID to attach a different transaction than the one CICS has decided to run (other R2 parameter list values must not be changed).
2. See "Chapter 4.1. ACF/VTAM logical units with CICS" on page 185 for operation of this exit.

XZCIN

Location	*DFHZCB*
Description	After a VTAM input event

Exit-specific registers

Register	**Addresses**
R8	TIOA (may be 0)
R10	TCTTE

Valid return codes
None

Processing information:

1. This exit is not taken for LU6.2 operations
2. See "Chapter 4.1. ACF/VTAM logical units with CICS" on page 185 for operation of this exit.

XZCOUT

Location	*DFHZCB*
Description	Before a VTAM output event

Exit-specific registers

Register	**Addresses**
R9	VTAM RPL
R10	TCTTE

Valid return codes
None

Processing information:

1. This exit is not taken for LU6.2 operations
2. See "Chapter 4.1. ACF/VTAM logical units with CICS" on page 185 for operation of this exit.

XZCOUT1

Location	*DFHZCB*
Description	Before a VTAM message is broken up into RUs

Exit-specific registers

Register	**Addresses**
R8	TIOA (may be 0)
R9	VTAM RPL
R10	TCTTE

Valid return codes
None

Processing information:

1. This exit is not taken for LU6.2 operations
2. See "Chapter 4.1. ACF/VTAM logical units with CICS" on page 185 for operation of this exit.

Chapter 5.2. Exit to allow modification and redirection of CICS messages

In an installation with many CICS systems, having automated responses to a limited number of frequently occurring messages can greatly increase efficiency. This is feasible if each message contains the applid of the originating system and if all messages are routed to a single point of control. However, in a typical large installation, messages go to a variety of transient data destinations, and many messages do not contain the applid of the originating system. In CICS/MVS 2.1.2, the XTDCOUT global user exit provides a solution to these problems.

The XTDCOUT global user exit

The XTDCOUT global user exit makes automated operation more practicable. The XTDCOUT exit is in the CICS transient data program DFHTDP, and allows a user program to modify and redirect messages before they are written to a CICS system queue.

An XTDCOUT exit program can:

- Redirect a message to the console
- Modify the text of a redirected message, for example, by adding the applid of the sending CICS system
- Add to a redirected message the applid of the sending CICS system
- Suppress a message.

Note: If you modify a message, never modify the original. Always copy the message before modifying it. If a message is not suppressed, the original, unmodified, version is written to the transient data queue.

Parameter list

On entry to the exit, the address of the following parameter list is in the register 1 field (offset 24 bytes) of the register save area addressed by field UEPHMSA.

Table 4. Parameter list for the XTDCOUT exit

OFFSET	LENGTH	CONTENTS
0	4	Queue name (see note 1)
4	4	Address of data (see notes 2 and 3)
8	4	Length of data (see notes 2 and 3)

Notes:

1. The queue name is the *real* name of the queue. For a destination defined with DFHTCT TYPE=INDIRECT, the name is the INDEST name.
2. For a record with a variable length record prefix, the address is the address of the data, and the length does not include the length of the prefix.
3. Do not modify the data; if a redirected message is to be modified, apply the modification to a copy of the original message.

Return codes

The XTDCOUT exit can return two codes, 0 and 4. If any other value is returned, 0 is assumed. The effects of the codes are as follows:

Table 5. Return codes for the XTDCOUT exit

RETURN CODE	MEANING
0	DFHTDP continues processing the message. When the message is for an intrapartition transient data destination, the XTDOUT exit is invoked if it is enabled. CICS writes the message to the specified queue.
4	DFHTDP returns control to the caller. No message is written to the queue.

Sample XTDCOUT exit program, DFHUXSP1

A sample XTDCOUT exit program DFHUXSP1 (see Figure 25 on page 331) is shipped with CICS. Briefly, DFHUXSP1:

1. Checks the message against the "Message Number/TD Queue action list," lines 17400-20400 (this determines if the message is to be suppressed or rerouted).
2. For a message to be rerouted, inserts the applid if necessary, writes the message to the console, and sets return code 4.
3. For a message to be suppressed, sets return code 4.
4. For a message not in the "Message Number/TD Queue action list," sets return code 0.

Customizing DFHUXSP1

DFHUXSP1 is designed to be used with minimum alteration, and is copiously commented to help you understand it.

The routine that you must tailor to your requirements is the "Message Number/TD Queue action list," lines 17400-20400. In the sample, the actions are:

- Suppress message number 5935 and all messages for the CSMT queue
- Redirect message number 3501 and all messages for the CSML queue.

The sample assumes the most likely case — that you do not want further processing by DFHTDP of a message that you have rerouted. If this is not what you want, change the return code setting accordingly.

The sample does not change the message text of a redirected message. If you wish to do this, the best place is in the applid-insertion routine.

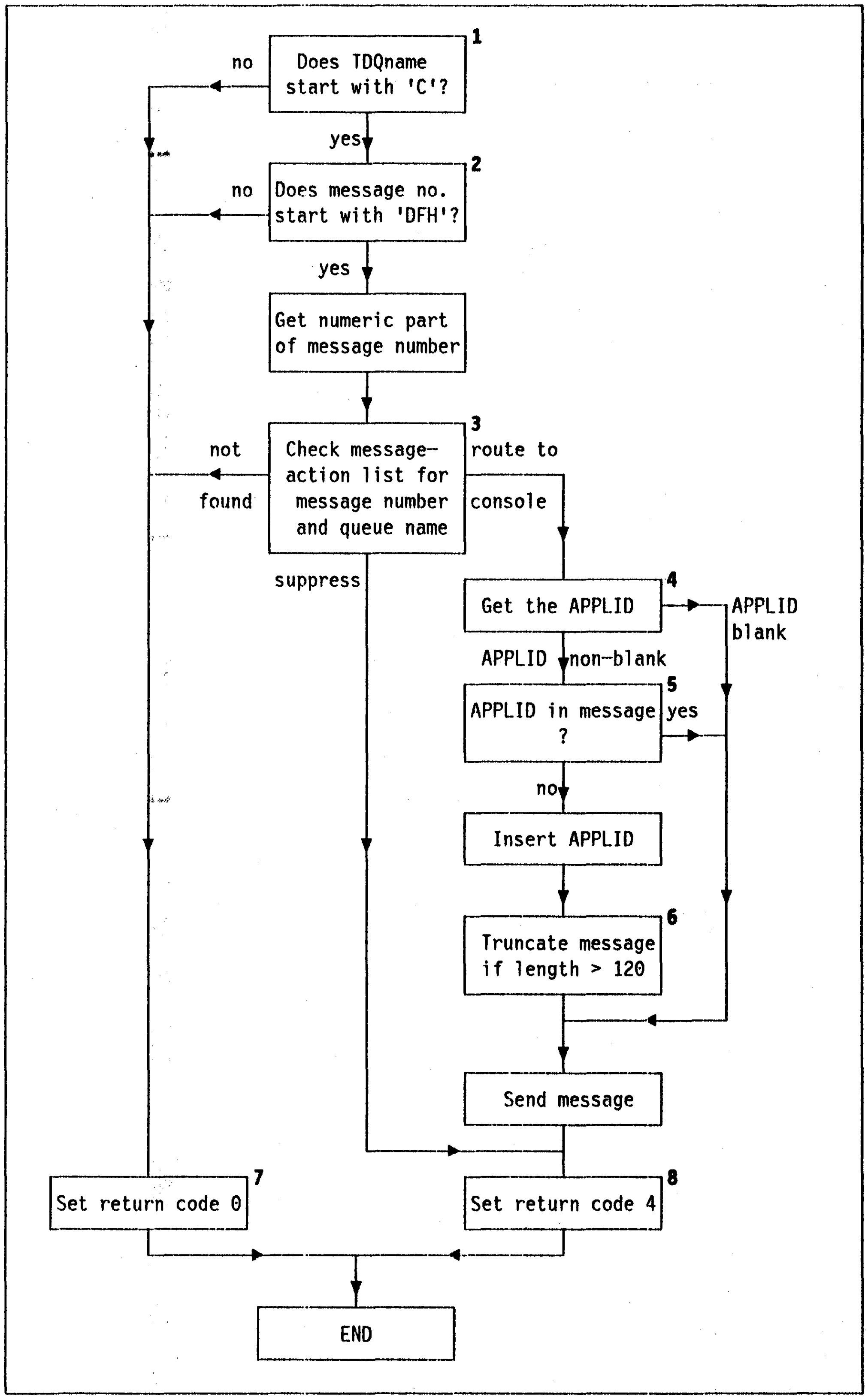
Indirect queue names

If the destination of a message is defined in the DCT as an *indirect queue name*, CICS passes the indirect queue name to the exit program for use in selecting messages for rerouting or suppression. If necessary, modify your DCT entries to obtain the results you want.

Logical flow of DFHUXSP1

Figure 24 on page 330 shows the logical flow of processing in DFHUXSP1. The following notes refer to the numbers in Figure 24.

1. Check that the queue is a CICS system queue, that is, the queuename starts with C.
2. Check that the message is a CICS message, that is, the message number starts with DFH.
3. Check whether the message is one that you want to reroute or suppress. The term "message-action list" is an abbreviation of "Message Number/TD Queue action list" and refers to the list of messages and queues for which you wish to take action.
4. The applid of your CICS system can be blank, in which case no applid can be inserted in the message.
5. Check if the message already contains the applid, in which case insertion is unnecessary.
6. Inserting the applid can make the message length exceed the maximum of 120 characters. If this occurs, truncate the message.
7. The exit program has taken no action for this message. With return code 0, the program signals DFHTDP to process the message normally, invoking exit XTDOUT (if enabled) and writing data to the system queue.
8. The exit program has suppressed or rerouted this message. With return code 4, the program signals DFHTDP to return control to the caller without invoking exit XTDOUT or writing data to the system queue.



1
Does TDQname start with 'C'?
no
yes
2
Does message no. start with 'DFH'?
no
yes
Get numeric part of message number
3
Check message-action list for message number and queue name
not found
route to console
suppress
4
Get the APPLID
APPLID blank
APPLID non-blank
5
APPLID in message ?
yes
no
Insert APPLID
6
Truncate message if length > 120
Send message
7
Set return code 0
8
Set return code 4
END

Figure 24. Logical flow of sample program for XTDCOUT exit

Listing of DFHUXSP1

```
*       This Sample is enabled by the CECI command:-              * 00002000
*       CECI EXEC CICS ENABLE EXIT(XTDCOUT) PROGRAM(DFHUXSP1) START  * 00002100
*       This Sample is disabled by the CECI command:-             * 00003700
*       CECI EXEC CICS DISABLE EXIT(XTDCOUT) PROGRAM(DFHUXSP1) STOP  * 00003800
*                                                                 * 00003900
*       The inputs to the sample are:-                            * 00004000
*       1.  The User Exit parameter list;-                        * 00004100
*           DFHUEPAR                                              * 00004200
*       2.  The CSA parameters;-                                  * 00004300
*           DFHCSADS                                              * 00004400
*       3.  The SIT dataset;-                                     * 00004500
*           DFHSIT                                                * 00004600
*                                                                 * 00004700
*       The outputs from the sample are:-                         * 00004800
*       1.  Return Code Zero;-                                    * 00004900
*           This instructs TD to continue processing this message.  * 00005000
*       2.  Return Code Four;-                                    * 00005100
*           This instructs TD stop further processing of this     * 00005200
*           message.                                              * 00005300
*                                                                 * 00005400
*********************************************************************** 00016800
*   DFHUXSP1 Sample User Exit for 2.1.2                           * 00016900
*********************************************************************** 00017000
*   Standard Equates                                              * 00017100
*********************************************************************** 00017200
R00      EQU 0                      Not Used                        00017300
R01      EQU 1                      Address of Exit Parameter List  00017400
R02      EQU 2                      Copy of Exit Parameter List, we are 00017500
*                                   using this register             00017600
R03      EQU 3                      Anchor register/scanning register 00017700
R04      EQU 4                      APPLID                          00017800
R05      EQU 5                      Work register for holding things 00017900
R06      EQU 6                      Work register to ADDR Exit Plist 00018000
R07      EQU 7                      Used for counting in search loops 00018100
R08      EQU 8                      Work register for APPLID/CSA    00018200
R09      EQU 9                      Work register for Message Text  00018300
R10      EQU 10                     Work register for Message Length 00018400
R11      EQU 11                     Used for base registers         00018500
R12      EQU 12                     Address of the TCA of invoking task 00018600
R13      EQU 13                     Address of the standard register 00018700
*                                   save area                       00018800
R14      EQU 14                     Return Address to DFHUEH for    00018900
*                                   branching back to calling module 00019000
R15      EQU 15                     Entry Address into the exit Program 00019100
*********************************************************************** 00019200
*   Equates for this Program                                      * 00019300
*********************************************************************** 00019400
MSG_ADDR EQU   R09                  Address of the Message Text     00019500
APPLID   EQU   R04                  Address of the APPLID           00019550
MSG_LEN  EQU   R10                  Address of the Message Text Length 00019600
```

Figure 25 (Part 1 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
         COPY  DFHCSADS            Look at the CSA Parameters
         DFHSIT TYPE=DSECT         Look at the SIT
         DFHUEXIT TYPE=EP          Provide DFHUEPAR parameter List File
*                                  File Control Program and List of
*                                  of EXITID equates
***********************************************************************
DFHUXSP1 CSECT  ,
DFHUXSP1 AMODE 31                  Execution is in 31-BIT Mode
*                                  Addressing
***********************************************************************
         SAVE  (14,12)             Save calling Program's registers
***********************************************************************
         LR    R11,R15             Set up User Exit Program's Base
         USING DFHUXSP1,R11        register
***********************************************************************
         LR    R02,R01             Set up Addressing for User Exit
         USING DFHUEPAR,R02        Parameter List
***********************************************************************
*   DFHUXSP1 Main Routine.                                            *
***********************************************************************
*        1. Check to see if the TD queue is a CICS queue.             *
*           If the Queue name begins with a 'C', then carry on        *
*           processing. If not, then set the return code to Zero      *
*           before Ending.                                            *
*        2. Check to see if the Message data begins with DFH.         *
*           If the Message Data begins with 'DFH', then carry on      *
*           processing. If not, then set the return code to Zero      *
*           before Ending.                                            *
*        3. Get the Message number from the Message data and          *
*           put it into MSGSNO.                                       *
*        4. CALL the MSG_TDQS routine to see what needs to be done.   *
***********************************************************************
*   Start the Sample Program Main Routine here.                       *
***********************************************************************
MAINPROG DS    0H
         L     R07,UEPHMSA                  Address of registers passed
*                                           from CICS
         L     R06,24(R07)                  Address of parameter list
***********************************************************************
CSA_ADDR DS    0H
         L     R08,UEPCSA                   Set up the Addressing for
         L     R03,CSASITBA-DFHCSADS(R08)   Address CSA SIT parameters
         LA    APPLID,SITSAPLD-DFHSITDS(R03)  Get the specific APPLID
         SR    R03,R03
```

Figure 25 (Part 2 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   Check to see if this is a CICS TD Message.                        *
***********************************************************************
FINDCICS DS    0H
         CLI   0(R06),C'C'             Check to see if the first..
*                                      character of the Queue Name
*                                      is 'C'
         BNE   RCNORMAL                If no, then set the return
*                                      code to Zero before Ending
*                                      When the return code is set
*                                      to Zero, CICS will continue
*                                      to process the Message.
***********************************************************************
*   Check to see if Message TEXT begins with DFH.                     *
***********************************************************************
FIND_DFH DS    0H
         L     MSG_ADDR,4(R06)         Get the Message Address
*                                      from the Parameter list
         CLC   0(3,MSG_ADDR),=C'DFH'   If the first 3 characters
*                                      of the Message Data are...
         BNE   RCNORMAL                ...not 'DFH' then End with
*                                      Return Code Zero.
***********************************************************************
*        Here we are trying to find where the Message Number starts.  *
*        We do this by scanning the first nine characters of the      *
*        Message Text until we find the first numeric character. We   *
*        do this by checking to see if each character in turn is      *
*        outside the range 'A' to 'Z'. If not then we know it is an   *
*        Alphabetic character.                                        *
*        We do this because we know that the Message Number has the   *
*        format 'DFHyyxxxx' or 'DFHxxxx', where 'yy' is the Domain-id *
*        and 'xxxx' is the Message Number.                            *
***********************************************************************
FINDMSGN DS    0H
         CLI   3(MSG_ADDR),C'0'        Check if less than '0'
         BL    RCNORMAL                If yes, then it is not a
*                                      message number
         CLI   3(MSG_ADDR),C'9'        Check if greater than '9'
         BH    RCNORMAL                If yes, then it is not a
*                                      message number
***********************************************************************
*         save message number                                         *
***********************************************************************
```

Figure 25 (Part 3 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   MSG_TDQS                                                          *
***********************************************************************
*        <<<<<< Start Of Section To Be Modified By User >>>>>>        *
*        <<<<<< Start Of Section To Be Modified By User >>>>>>        *
*        <<<<<< Start Of Section To Be Modified By User >>>>>>        *
***********************************************************************
*        Check each Message Number/TD Queue Name, and Suppress using  *
*        RCBYPASS or Re-Route. If Re-Route then Call the CHECK_ID     *
*        routine before Ending with RCBYPASS. If its neither Suppress *
*        or Re-Route then Call RCNORMAL before Ending.                *
***********************************************************************
*   MESSAGE NUMBER LIST.                                              *
***********************************************************************
MSG_TDQS DS    0H
         CLC   3(4,MSG_ADDR),=C'5935'      Is MSGSNO equal to '3501'
         BE    RCBYPASS                    If yes, then branch to...
*                                          ...RCBYPASS then End.
***********************************************************************
         CLC   0(4,R06),=C'CSMT'           Is TDQNAME equal to 'CSMT'
         BE    RCBYPASS                    If yes, then branch to...
*                                          ...RCBYPASS then End.
***********************************************************************
         CLC   3(4,MSG_ADDR),=C'3501'      Is MSGSNO equal to '3501'
         BE    GET_APPL                    If yes,then branch to the..
*                                          ...GET_APPL routine.
***********************************************************************
         CLC   0(4,R06),=C'CSML'           Is TDQNAME equal to 'CSML'
         BE    GET_APPL                    If yes,then branch to the..
*                                          ...GET_APPL routine.
         B     RCNORMAL                    If no,MSGSNO/TDQNAME not in
*                                          the action list, therefore
*                                          branch to RCNORMAL then End
***********************************************************************
*        <<<<<< End Of Section To Be Modified By User >>>>>>          *
*        <<<<<< End Of Section To Be Modified By User >>>>>>          *
*        <<<<<< End Of Section To Be Modified By User >>>>>>          *
***********************************************************************
***********************************************************************
*   Get the APPLID to start with.                                     *
***********************************************************************
GET_APPL DS    0H
         LA    R07,0                       Set Counter to zero
         LR    R03,APPLID                  Set R03 as scanning REG.
*                                          register 4 is used to step
*                                          the R07.
         LA    R05,8                       Put 8 into register 5
*                                          register 5 is used as the
*                                          R07 limit.
```

Figure 25 (Part 4 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   Check to see if there is an APPLID to be inserted.                *
***********************************************************************
NO_APPL  DS    0H
         CLC   0(8,R03),=C'        '         If the APPLID is blank,
*                                            then there is no APPLID
*                                            present to be inserted, so
*                                            we can send the Message
         BE    SENDSMSG                      using a basic WTO
***********************************************************************
*   Get the length of the APPLID.                                     *
***********************************************************************
FINDAPPL DS    0H
         CLI   0(R03),C' '                   Check APPLID for a blank
         BE    CHECK_ID                      If yes, then we have found
*                                            the length of the APPLID
*                                            so now we can check the
*                                            Message text for the APPLID
         LA    R07,1(,R07)                   Add 1 to the R07
         LA    R03,1(,R03)                   Add 1 to register 3
*                                            By adding 1 to the scanning
*                                            register we are moving
*                                            along the Message Text so
*                                            we can scan the next
*                                            character.
         CR    R07,R05                       We check register 5 to see
*                                            if we have reached the
*                                            maximum length of APPLID.
         BNH   FINDAPPL                      If no, then we must try the
*                                            next character.
***********************************************************************
*   CHECK_ID                                                          *
***********************************************************************
*   Check to see if there is an APPLID in the Message Text.           *
***********************************************************************
*        In this routine we are checking to see if the Message Text   *
*        already contains the APPLID. We do this by scanning starting *
*        at the first character of the Message Text for a length      *
*        which is equal to the length of the APPLID.                  *
*        Having done the first check, we move to the next character   *
*        and repeat the check.                                        *
*        We continue this checking until we find the APPLID or we     *
*        Reach the maximum length of the Message Text. The length of  *
*        the Message Text is held in MSG_LEN.                         *
*        If the APPLID is found we can just send the Message using    *
*        the SENDSMSG routine. The SENDSMSG routine does a basic WTO  *
*        before setting the return code to Four then Ending the       *
*        Sample.                                                      *
```

Figure 25 (Part 5 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
CHECK_ID DS    0H
***********************************************************************
*   Get the length of the Message Text.                               *
***********************************************************************
         L     MSG_LEN,8(R06)               Get the Message Length
*                                           from the Parameter list
***********************************************************************
*   Set up the loop for searching the Message Text.                   *
***********************************************************************
START_LP DS    0H
         LR    R05,R07                      Put the length of the
*                                           APPLID into R05
         LA    R07,0                        Set the counter to Zero
         LR    R03,MSG_ADDR                 Set R03 as scanning reg.
*                                           register 4 is used to step
*                                           the R07.
*        SR    MSG_LEN,R05                  SUBTRACT THE APPLID LENGTH
*                                           from the Message length
*        LA    R08,APPLID                   Save the address of the
*                                           APPLID.
***********************************************************************
*   The Message Text search loop starts here.                         *
***********************************************************************
APPLFIND DS    0H
         BCTR  R05,0                        Reduce R05 before EXecute
         EX    R05,CHK_APPL                 EXecute instruction used
*                                           to help find the APPLID
         LA    R05,1(,R05)                  Increase R05 by one after
*                                           EXecute(this will not alter
*                                           the condition code from the
*                                           EXecute instruction).
         BE    SENDSMSG                     If the APPLID is present in
*                                           the Message text we can now
*                                           send the Message.
         LA    R07,1(,R07)                  Add 1 to the R07
         LA    R03,1(,R03)                  Add 1 to register 3
*                                           By adding 1 to the scanning
*                                           register we are moving
*                                           along the Message Text so
*                                           we can scan the next
*                                           character.
         CR    R07,MSG_LEN                  Have we reached the Maximum
*                                           Length of the Message Text
         BNH   APPLFIND                     If no, then check the next
*                                           character.
*                                           If yes, then the APPLID is
*                                           not present in the Message
*                                           Text, so we have to insert
*                                           the APPLID into the Message
*                                           Text using the INSERTID
*                                           routine.
```

Figure 25 (Part 6 of 11). Sample program for XTDCOUT exit

```
************************************************************************
*   INSERTID                                                           *
************************************************************************
*        Insert the APPLID in between the Message Number and the rest *
*        of the Message Text.                                          *
************************************************************************
*   Set up the loop for searching the Message Text to find the first  *
*   Blank after the Message Number.                                    *
************************************************************************
INSERTID DS    0H
         LA    R07,0                      Set the counter to Zero
         LR    R03,MSG_ADDR               Set R03 as scanning reg.
*                                         register 4 is used to step
*                                         the R07.
         L     MSG_LEN,8(R06)             Get the Message Length
*                                         from the Parameter list
************************************************************************
*   The Message Text search loop starts here.                          *
************************************************************************
FIND_BNK DS    0H
         CLI   0(R03),C' '                Check Message Text for a
*                                         Blank(this cannot be the
*                                         first character as we have
*                                         already checked the first
*                                         of the Message text in the
*                                         FIND_DFH routine at the
*                                         start of the Sample).
         BE    INS_APPL                   The Blank has been found so
*                                         R07 will now hold its
*                                         offset
         LA    R07,1(,R07)                Add 1 to the R07
         LA    R03,1(,R03)                Add 1 to register 3
*                                         By adding 1 to the scanning
*                                         register we are moving
*                                         along the Message Text so
*                                         we can scan the next
*                                         character.
         CR    R07,MSG_LEN                Have we reached the Maximum
*                                         Length of the Message Text
         BNH   FIND_BNK                   If no, then check the next
*                                         character.
         B     RCNORMAL                   If yes, then no Blank could
*                                         be found in the Message
*                                         Text, therefore the Message
*                                         Text must be in error, so
*                                         End the Sample with a
*                                         return code of Zero. This
*                                         will have the effect that
*                                         the Message will be passed
*                                         unchanged.
```

Figure 25 (Part 7 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   Now insert the APPLID after the Message Number                    *
***********************************************************************
INS_APPL DS    0H
         LR    R03,MSG_ADDR                Set R03 as Message Text
*                                          work register
         LA    R06,MOD_TEXT                Set R06 to Address the
*                                          output area.
         BCTR  R07,0                       Reduce R07 before the
*                                          EXecute instruction.
*                                          (R07 will always be
*                                          greater than Zero because
*                                          the first blank in the
*                                          Message Text appears after
*                                          Message Number. see the
*                                          FIND_BNK routine)
         EX    R07,MOVE_NUM                Use the EXecute instruction
*                                          to move the Message Number
*                                          into the output area (the
*                                          Message Number has the form
*                                          DFHyyyy, where yyyy is the
*                                          MSG Number).
         LA    R07,1(,R07)                 Increase R07 by one
*                                          after EXecute
         ALP   R06,R07                     Add to R06 the length of
*                                          Message Number.
         LA    R06,1(,R06)                 Add 1 to R06 to increase
*                                          the offset in readiness for
*                                          APPLID move.
         BCTR  R05,0                       Reduce R05 before EXecute
*                                          (R05 contains the true
*                                          length of the APPLID).
         EX    R05,MOVE_APL                Use the EXecute instruction
*                                          to move the APPLID into the
*                                          output area.
         LA    R05,1(,R05)                 Increase R05 by one after
*                                          EXecute.
         ALR   R06,R05                     Add the length of the
*                                          APPLID to R06(R06 contains
*                                          the address of the output
*                                          area).
         ALR   R03,R07                     Increase the offset in R03
*                                          by the length of the
*                                          Message Number.
         LA    MSG_LEN,1(,MSG_LEN)         Add one to the message text
*                                          length to allow for the
*                                          blank that will be INSERTED
*                                          into the output message
*                                          text(after the APPLID).
```

Figure 25 (Part 8 of 11). Sample program for XTDCOUT exit

```
         ALR   MSG_LEN,R05                 Add the length of the       00054200
*                                          APPLID in R05 to the Max    00054300
*                                          length of the output        00054400
*                                          message text.               00054500
         BCTR  MSG_LEN,0                   Reduce MSG_LEN before the   00054600
*                                          EXecute.                    00054700
         EX    MSG_LEN,MOVEREST            Use the EXecute instruction 00054800
*                                          to move the rest of the     00054900
*                                          message text into the       00055000
*                                          output declared by MOD_TEXT 00055100
         LA    MSG_LEN,1(,MSG_LEN)         Increase MSG_LEN by one     00055200
*                                          after EXecute.              00055300
*********************************************************************** 00055500
*   CHECKLEN                                                          * 00055600
*********************************************************************** 00055700
*        Check to see if the length of the Outputted Message Text will* 00055800
*        be greater than 120, if it will be than reduce the Message   * 00055900
*        Text length by that amount.                                  * 00056000
*********************************************************************** 00056100
CHECKLEN DS    0H                                                      00056200
         CH    MSG_LEN,CONST120            Check to see if the output  00056300
*                                          message text length is      00056400
*                                          greater than the maximum    00056500
*                                          allowed length of 120.      00056600
         BNH   SENDSMSG                    If the output length is     00056700
*                                          less than 120, then send    00056800
*                                          the message.                00056900
*********************************************************************** 00057100
*   TRUC_LEN                                                          * 00057200
*********************************************************************** 00057300
*        Truncate the Length of the Output Message Text to 120.       * 00057400
*********************************************************************** 00057500
TRUN_LEN DS    0H                          If the Output Message Text  00057600
*                                          length is greater than 120, 00057700
*                                          we will have to reduce the  00057800
*                                          Message Text length by how  00057900
*                                          much the Output Message     00058000
*                                          Text length exceeds 120.    00058100
         LH    MSG_LEN,CONST120            If the output message       00058200
*                                          length is greater than      00058300
*                                          120, then set the length to 00058400
*                                          120.                        00058500
```

Figure 25 (Part 9 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   SENDSMSG                                                          *
***********************************************************************
*        Do a basic WTO to the Console, branch to RCBYPASS before     *
*        Ending.                                                      *
***********************************************************************
SENDSMSG DS    0H
         LA    R06,TEXT_WTO                Address the output area
         LA    MSG_LEN,4(,MSG_LEN)         Add 1 to the Message Length
         STH   MSG_LEN,0(0,R06)            Store the Length in the
*                                          first two Bytes.
         WTO   MF=(E,(R06))
         B     RCBYPASS
***********************************************************************
*   RCNORMAL will set the return code to Zero, which tells TD to      *
*            continue processing.                                     *
***********************************************************************
RCNORMAL DS    0H
         LA    R15,0                       Set the Return Code to Zero
         B     END_MAIN
***********************************************************************
*   RCBYPASS will set the return code to Four, which tells TD to      *
*            stop TD processing.                                      *
***********************************************************************
RCBYPASS DS    0H
         LA    R15,4                       Set the Return Code to Four
         B     END_MAIN
***********************************************************************
*   END_MAIN will end the sample and pass back to the caller the      *
*            return code.                                             *
***********************************************************************
END_MAIN DS    0H
         L     R13,UEPEPSA
         RETURN (14,12),RC=(15)
***********************************************************************
*   Declarations                                                      *
***********************************************************************
CONST120 DC    H'120'                      This is the constant for
*                                          the maximum length of the
*                                          Output Message.
TEXT_WTO WTO   '                                                       X
                                                                       X
                       ',MF=L              Max text length 120 Chars.
MOD_TEXT EQU   TEXT_WTO+4                  Text work area
```

Figure 25 (Part 10 of 11). Sample program for XTDCOUT exit

```
***********************************************************************
*   Execute Instructions                                              *
***********************************************************************
         DS    0H                            To align for the EXecute
*                                            instruction
CHK_APPL CLC   0(0,R03),0(R04)               Used with EXECUTE instruct-
*                                            -ion/for finding the APPLID
*                                            in the Message Text
         DS    0H                            To align for the EXecute
*                                            instruction
MOVE_NUM MVC   0(0,R06),0(R03)               Moving Message Number to
*                                            work area.
         DS    0H                            To align for the EXecute
*                                            instruction
MOVE_APL MVC   0(0,R06),0(APPLID)            Moving APPLID into the
*                                            work area
         DS    0H                            To align for the EXecute
*                                            instruction
MOVEREST MVC   0(0,R06),0(R03)               Moving the rest of the
*                                            Text to work area
         LTORG ,
         END DFHUXSP1
```

Figure 25 (Part 11 of 11). Sample program for XTDCOUT exit

Chapter 5.3. File control status exits

Exits XFCSREQ and XFCSREQC are driven respectively before and after the status of a file is changed (by CEMT or by EXEC SET FILE). The R6 request byte is provided in both exits, but the R7 response byte is only available for XFCSREQC.

The exits can be driven more than once for each file. This is because there can be multiple users of a file. For example, if a CEMT SET FILE CLOSE is executed, the exits are driven for the CEMT and for the last task using the file.

Request byte

The request byte can be mapped via the User Exit macro. Here are the settings:

R6 addresses XL1	Operation
UEPFSCLS	CLOSE the file
UEPFSDIS	DISABLE the file
UEPFSENB	ENABLE the file
UEPFSOPN	OPEN the file.

Rejecting the request

XFCSREQ can reject the request, by returning 4 in R15. This leaves the status of the file unchanged, except for the an OPEN request on a file that is CLOSED,ENABLED. In this case, the file status is changed to CLOSED,UNENABLED. This caters for the possibility that an implicit open is being run, and ensures that an application program receives a NOTOPEN response to a subsequent request. If the request is rejected, then DFH0996 is issued, and XFCSREQC is driven with a Failed Response.

Response byte

The Response byte can be mapped via the User Exit macro. Here are the settings:

R7 addresses XL1	What happened
UEFSFAIL	The request failed
UEFSNORM	The request completed normally
UEFSPEND	The request generated a PENDING response
UEFSWARN	The request completed OK, but a Warning message was issued.

PENDING response

The PENDING response is generated by a CLOSE operation, if active tasks are still using the file.

Consider these two CLOSE requests, with multiple tasks using the file:

1. A CLOSE NOWAIT is issued. The request completes, but invokes XFCSREQC with a PENDING response, as tasks are still using the file. When all these tasks complete, the last task closes the file and drives XFCSREQ and XFCSREQC.
2. A CLOSE WAIT is issued. The requesting task drives XFCSREQ and enters a WAIT state. One by one, the other tasks using the file terminate, the last task driving the XFCSREQ and XFCSREQC exits as it closes the file. The waiting task now resumes execution, and drives XFCSREQC without a PENDING response, because the file is now closed.

Chapter 5.4. Task-related user exits

General-Use Programming Interface

This chapter describes a special kind of user exit called a **task-related user exit**. A task-related user exit allows you to write your own program to access a recoverable resource, such as a database, that would not otherwise be available to your CICS system. Such a resource is known as a non-CICS resource. The exit is said to be task-related because it becomes part of the task that invoked it and because, unlike a global user exit, it is not associated with an exit point. Three CICS management services (syncpoint manager, monitoring and task manager) may invoke a task-related user exit.

Introduction to the task-related user exit mechanism (the adapter)

The task-related user exit mechanism is known as an 'adapter' because it provides the connection between an application program that needs to access a non-CICS resource and the manager of that resource. Figure 26 on page 346 illustrates the adapter concept.

The adapter is made up of three or more locally-written programs. These are a 'stub', a task-related user exit program, and one or more administration routines or programs.

The **stub** intercepts a request (for example, to access a non-CICS resource) that is issued by the calling application program. The stub can be used to resolve a locally-defined high-level language command into a task-related user exit macro call, DFHRMCAL, which then causes CICS to pass control to the task-related user exit program.

The **task-related user exit program** changes commands for accessing a non-CICS resource into a form acceptable to the resource manager. The program is written in assembler language. It is executed in response to a specific application program request, for example, to access a resource. It may be passed application data, such as a search argument for a required record. Responses from the resource manager are passed back to the calling program by the task-related user exit program.

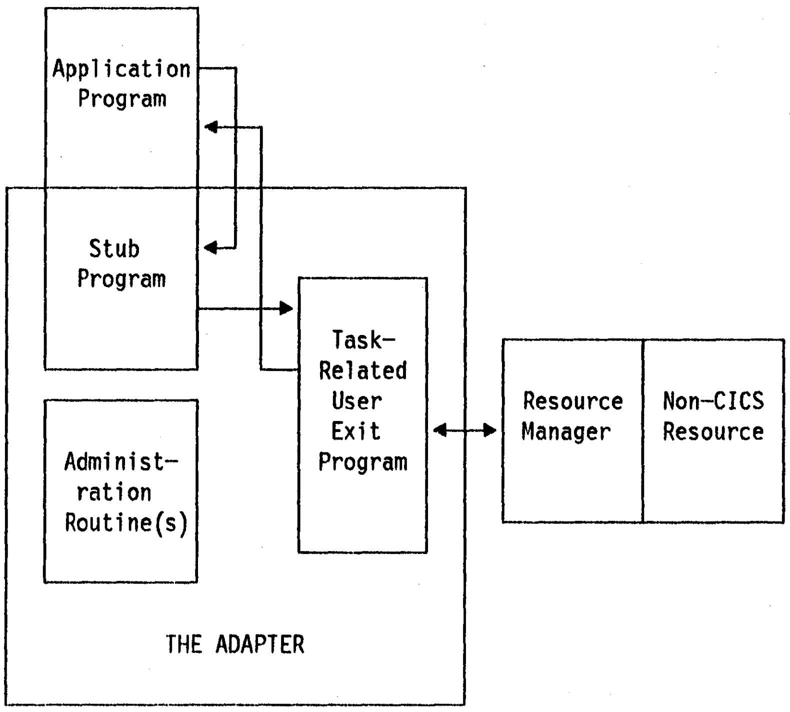


Figure 26. The adapter concept

The task-related user exit program is provided with a parameter list by the CICS management module that handles task-related user exits. This parameter list (DFHUEPAR) gives the exit program access to information such as the addresses and sizes of its own work areas.

The task-related user exit program may be invoked by the CICS task manager, the CICS syncpoint manager and/or CICS monitoring, as well as by an application program. The parameter list serves to distinguish among these various callers. It also gives access to a register save area containing the caller's registers.

The **administration routine(s)** contain the EXEC CICS ENABLE and DISABLE commands that you use to install and withdraw the task-related user exit program. The administration routines may also contain commands to retrieve information about one of the exit program's work areas (the EXEC CICS EXTRACT EXIT command), and to resolve any inconsistency between CICS and a non-CICS resource manager after a system failure (the EXEC CICS RESYNC command).

The remainder of this chapter discusses each of these parts of the adapter in turn. For a description of how CICS handles task-related user exits, see the *CICS/MVS Diagnosis Reference* manual.

The stub program

The purpose of the stub program is to shield your application programmers from the mechanics of non-CICS resource managers. It is written in assembler language. After assembly, the stub is link-edited to each application program that wants to use it, and it has the following format:

```
          ENTRY    statname
            .
            .
statname  DFHRMCAL TO=ename
            .
            .
          END
```

statname is a label that can be referenced externally. It should conform to the requirements of an assembler-language ENTRY statement, and typically resolves a V-type address constant, or the target of a high-level language CALL. A single stub may contain several such labels.

ename is the entry name (specified on the EXEC CICS ENABLE command) of the task-related user exit program that you want to handle resource manager requests. This entry name must have been enabled and started before it is called for the first time.

You can define high-level language commands that your programmers will use when they want to access a non-CICS resource. You will need a translator to convert a locally-defined high-level language command into a conventional CALL to the required entry point of the stub program. Alternatively, the application program can issue a CALL naming the stub entry point. For example, your application programmer wishes to read a record from a non-CICS resource. He issues the COBOL command

```
CALL 'XYZ' USING PARM1 PARM2...
```

XYZ is an entry point (the statname) in your stub program. The stub converts the command into a macro call (DFHRMCAL) to the task-related user exit program, specified in the TO= operand, that handles resource manager requests. Return from the task-related user exit program is to the calling application program, not to the stub program.

A macro-level assembler program must place the address of a register save area in register 13 before invoking the stub. The program must restore the CSA address in register 13 on return from the stub. This is done automatically for macro-level COBOL and PL/I programs if a high-level language call is used to invoke the stub. For command-level programs, it is not necessary to save and restore register 13.

Returning control to the application program: If you specify RTNABND=YES in the DFHRMCAL macro, control returns to the application program when the task-related user exit is not available (because, for example, it is not enabled).

If you do not specify RTNABND=YES and the task-related user exit is not available, the application program terminates abnormally with the abend code AEY9.

Note for assembler language programs: A negative value in register 15 signals to the application program that control has returned because the exit is not available. The task-related user exit program can use positive values (including zero) in register 15 to pass resource manager response codes to the application program.

Task-related user exits and EDF: You can use the command-level execution diagnostic facility (EDF) to debug an application program that contains locally-defined high-level language commands. You can also use EDF to debug most task-related user exit programs that contain command-level statements. However, you cannot use EDF to debug exit programs that are invoked by CICS Monitoring, and these should be compiled with the NOEDF option.

If you define a high-level language for application programming, your translator generates, for each task-related user exit request in the application program, a call that is satisfied by the (DFHRMCAL) stub. The EDF message will refer to the name specified by the TO= operand of the generated DFHRMCAL macro. EDF will not be able to interpret the high-level statement for which it was generated.

For EDF to be able to display the parameter list, the calling program's register 1 must point to a list of addresses, and the high order bit must be set on to indicate the last address.

The task-related user exit program

The main function of the task-related user exit program is to change the calling program's parameters into a form acceptable to your non-CICS resource manager, and then to pass control to the resource manager. You will therefore need to be familiar with your resource manager's syntax requirements. The calling program's parameters are described beginning on page 351.

This section describes the user exit parameter lists, the schedule flag word, which is used by the exit program to register its need to be invoked by CICS management services, and register-handling in the task-related user exit program. This section also discusses the use of the CICS syncpoint manager and of the CICS task manager. Some notes about inappropriate actions are included.

User exit parameter lists

When a task-related user exit program is invoked, the CICS management module that handles task-related user exits provides the exit program with a parameter list that gives access to the following information:

- The identity of the calling program
- Addresses and sizes of any work areas that are available to the task-related user exit program
- The address of the register save area of the calling program

- The address of an EXEC interface block (EIB) that is for use by the task-related user exit program during this invocation
- The address of the identifier of the current unit of recovery
- The address of the schedule flag word.

To enable your exit program to access this parameter list you must include in it the macro instruction:

```
DFHUEXIT TYPE=RM
```

The DFHUEXIT TYPE=RM macro instruction causes the assembler to create the storage definitions (DSECTs) DFHUEPAR and DFHUERTR. The format and the purpose of these definitions are described below. The DSECTs are summarized in Figure 27 on page 353.

DFHUEPAR

DFHUEPAR gives you the following symbolic names for address parameters:

UEPEXN	Address of the function definition, which tells the task-related user exit program why it is being called. See "DFHUERTR (the function definition)" on page 350 for more details.
UEPGAA	Address of the global work area requested in the EXEC CICS ENABLE command. The global work area is described on page 357. CICS initializes this work area to X'00' when the task-related user exit program is enabled.
UEPGAL	Address of a halfword containing the length (binary value) of the global work area.
UEPTCA	Address of the TCA.
UEPCSA	Address of the CSA.
UEPHMSA	Address of the register save area (RSA) of the program containing the original call. (This is typically an application program, but can be the syncpoint manager, the CICS task manager or CICS monitoring.) It is an 18-word save area, with the contents of registers 14 through 12 stored in the fourth and subsequent words. Its fifth word, representing the calling program's register 15, is cleared by CICS before the task-related user exit program is invoked so that it can be used to convey response codes from the resource manager to the calling program. For this reason you cannot use register 15 to send data to the task-related user exit program. The seventh word of the save area contains the caller's register 1, which addresses the caller's parameter list when the caller is the CICS task manager, the CICS syncpoint manager, or CICS monitoring. When the caller is an application program, the contents of register 1 are determined by the linkage conventions of the adapter's language interface.
UEPTAA	Address of the local work area requested in the EXEC CICS ENABLE command. The local work area is described on page 357. CICS initializes the work area to X'00' throughout on first acquiring the area; that is, when the task first invokes the task-related user exit program.

UEPTAL Address of a halfword containing the binary

UEPEIB Address of the EXEC interface block (EIB) created by CICS for the task-related user exit program. The EIB exists only for the duration of the call and it allows the task-related user exit program to request CICS services through the command-level interface. Be aware that this is **not** the EIB that is available to the calling program, so you cannot access the calling program's environment other than by UEPHMSA (see above) which provides the address of the calling program's register save area (RSA).

UEPURID Address of CICS unit of recovery identifier. This is an eight-byte field that identifies the current logical unit of work.

UEPFLAGS Address of the schedule flag word. This is a fullword that the task-related user exit program uses to register its need for CICS management programs' services. For more information, see "The schedule flag word" on page 354.

DFHUERTR (the function definition)

The function definition identifies the caller of the task-related user exit program. The DSECT contains two symbolic definitions (fields).

UERTFGP This is a single byte that is set to X'00'. The zero setting shows that this is a task-related user exit invocation and that the parameter list therefore includes the fields UEPTAA, UEPTAL, UEPEIB, UEPURID and UEPFLAGS.

UERTFID This is a single-byte identifier that shows whether this call has been made by an application program, the CICS syncpoint manager, CICS monitoring or the CICS task manager. It can have one of the following four settings:

UERTAPPL	(X'02') indicates that the calling program is an application program
UERTSYNC	(X'04') indicates that the calling program is the syncpoint manager
UERTMONI	(X'06') indicates that the calling program is CICS monitoring (see "User exits for accessing monitoring data" on page 416)
UERTTASK	(X'08') indicates that the calling program is the CICS task manager.

It is important to know which type of program has made the call because it affects how the calling program's parameter list is interpreted by the task-related user exit program.

Caller parameter lists

In addition to the DSECTs DFHUERTR and DFHUEPAR, the inclusion of DFHUEXIT TYPE=RM in the task-related user exit program provides some field definitions that are specific to the program invoking the task-related user exit. The calling program's parameter list is normally addressed by R1 in the calling program's RSA, which is addressed by the field UEPHMSA of DFHUEPAR.

Application program parameters: If the caller is an application program, the format and addressing of its parameter list will be decided locally.

CICS syncpoint manager parameters: The first (and possibly the only) entry of the CICS syncpoint manager's parameter list is a pointer to a one-byte operation code. The code can be any valid combination of the following bit settings, each of which represents a syncpoint event.

UERTPREP	(X'80') Prepare to Commit.
UERTCOMM	(X'40') Commit Unconditionally.
UERTBACK	(X'20') Backout.
UERTDGCS	(X'10') Unit of recovery is lost to CICS cold start.
UERTDGNK	(X'08') Adapter should not be in doubt about this unit of recovery.
UERTLAST	(X'01') There will be no further units of recovery associated with this task. Note that when this bit is NOT set, there may or may not be further units of recovery. For this reason, it is not recommended that you rely on this bit to signal end-of-task. You should instead schedule the CICS task manager to drive you at end-of-task by setting the task manager bit in the schedule flag word. If you do use UERTLAST to signal end-of-task, and if at that stage you can complete your clean-up process, you can set the task manager bit off in the schedule flag word when the clean-up process is finished to avoid an unnecessary invocation by the CICS task manager.

The only **valid bit combinations** are the four produced by combining one of UERTCOMM, UERTBACK, UERTDGCS, and UERTDGNK, with UERTLAST. UERTPREP is not combined with any other setting.

If the operation code contains the bit settings UERTCOMM or UERTBACK, the parameter list may contain further entries. The additional parameters exist if the CICS syncpoint manager call is prompted by the issue of an EXEC CICS RESYNC command after a session or system failure. The EXEC CICS RESYNC command and the completion of the syncpointing procedure following a system failure are described in the section "Restart resynchronization" on page 365.

The additional parameters identify the task, the transaction that started the task, the terminal from which it was initiated, the identity of the terminal operator, and the date and time of the failing syncpoint. The last address in the parameter list is indicated by having its high-order bit set on. Typically, you would use these values to create meaningful messages for resource recovery. They are presented explicitly because, after a system failure, the task driving the exit is not the task that originally scheduled the recoverable work. These additional parameters describe the **original** task's environment and are accessed directly. Their format is given below.

2nd	Task	PL4
3rd	Tran	CL4
4th	Term	CL4
5th	Opid	CL4
6th	Date	PL4(00yyddd+)
7th	Time	PL4(0hhmmss+)

CICS task manager parameters: There is only one entry in the CICS task manager's parameter list. It addresses a single byte with bit definitions indicating the reason for the call.

UERTSOTR (X'40') Start of CICS task

UERTEOTR (X'80') End of CICS task

The schedule flag word should be set during the start-of-task call if you want your task-related user exit program to be invoked unconditionally by CICS monitoring or the CICS syncpoint manager.

CICS monitoring parameters: These are described in "User exits for accessing monitoring data" on page 416.

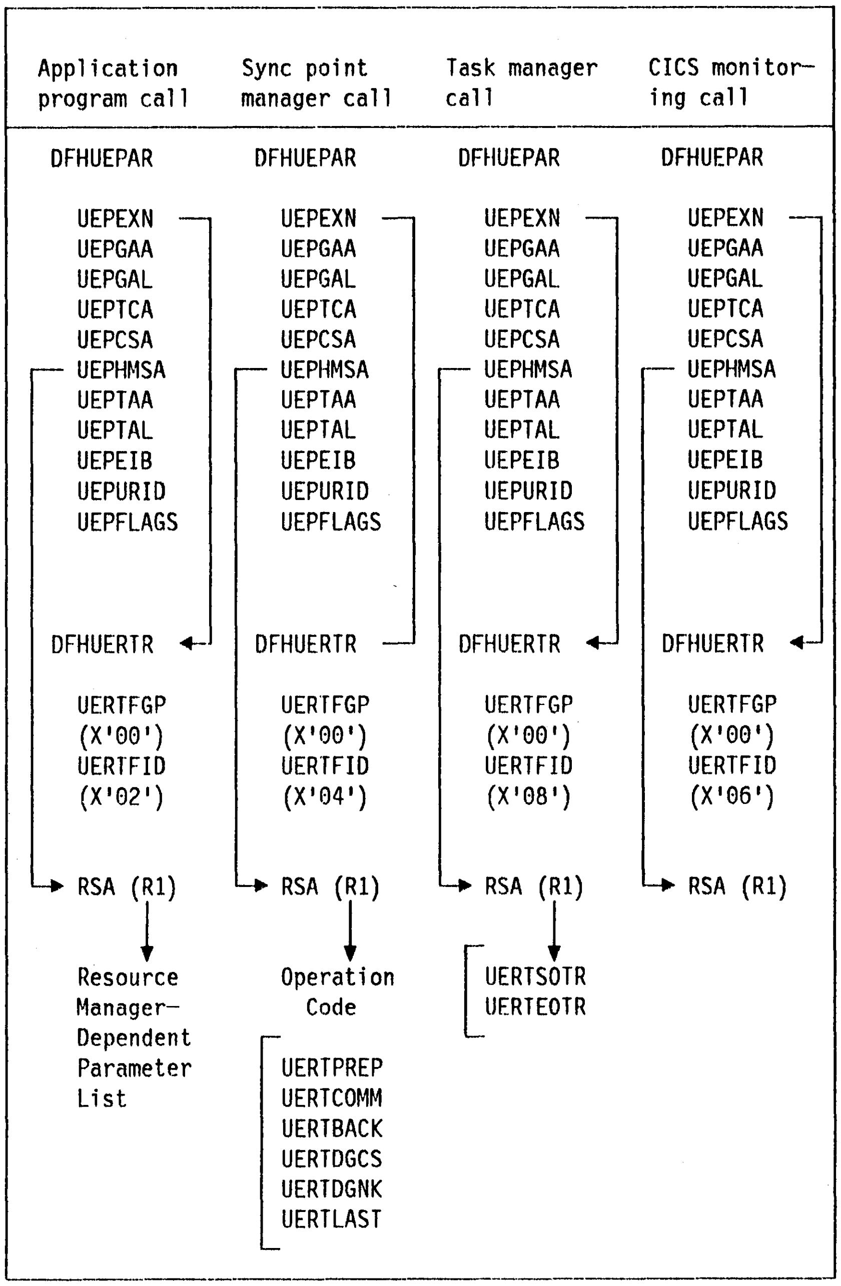


Figure 27. User exit parameter lists

Note: The CICS monitoring parameter list is described in "User exits for accessing monitoring data" on page 416.

The schedule flag word

The schedule flag word is a fullword indicator that the task-related user exit program uses to control its own invocation by the three available CICS management services. It is also used by CICS to schedule the first invocation of a task-related user exit program. The schedule flag word is accessed by the address parameter UEPFLAGS of DFHUEPAR. There is a unique schedule flag word for each association between a CICS task and the ENTRYNAME specified when a task-related user exit program is enabled. The format of the schedule flag word is shown in Figure 28.

Byte	Setting	Comments
0		Reserved
1		Reserved
2		
UEFDTASK	UEFMTASK (X'01')	Bit mask for task manager exit.
3		
UEFPMONI	UEFMMONI (X'40')	Bit mask for monitoring exit.
UEFDSYNC	UEFMSYNC (X'10')	Bit mask for syncpoint manager exit.
UEFDAPPL	UEFMAPPL (X'04')	Bit mask for API exit.

Figure 28. Format of the schedule flag word

The bit settings of the schedule flag word show which programs will invoke your task-related user exit program. For example, if an exit program is to be invoked by the CICS task manager, CICS monitoring, the CICS syncpoint manager **and** an application program, the last two bytes of the schedule flag word will be set to X'0154'. If an exit program is to be called by the CICS task manager and an application program only, the last two bytes of the flag word will be set to X'0104'. Before the exit program is first called by a task, CICS sets on the API flag bit. If you set the task manager bit on, either by specifying TASKSTART on the EXEC CICS ENABLE command, or by setting it from within the exit program, your exit program will be called at the start and finish of every task.

Before returning from the first call, or any subsequent call, the exit program can change the bit settings of the flag word to register its need to be invoked by a different CICS management service, or to register lack of interest in a service by setting the relevant flag bit to zero.

For example, an exit program may be called by an application program that needs to access a non-CICS recoverable resource. When the exit program is first called, the API bit is set on by CICS. If the calling program then issues a request to update a record, the exit program sets the syncpoint manager bit on in the schedule flag word. When the calling application program subsequently issues a syncpoint command, or when end-of-task is reached, the CICS syncpoint manager calls the exit program.

Note: CICS sets the syncpoint manager bit off after every call to the syncpoint manager. This is to avoid the CICS syncpoint manager invoking the task-related user exit program for a unit of recovery during which the exit program did no recoverable work. The syncpoint manager bit must therefore be set on whenever the exit program performs any recoverable work.

Register-handling in the task-related user exit program

In this section two sets of registers are discussed:

- The first set is the registers belonging to the CICS management module that handles task-related user exits. These are referred to as the **CICS registers**.
- The second set is the registers belonging to the calling program, which may be an application program, the CICS syncpoint manager, the CICS task manager or CICS monitoring. These are referred to as the **calling program's registers**.

Saving CICS registers

Your task-related user exit program should begin by saving the contents of the CICS registers. Register 13 addresses an 18-word area into whose 4th and subsequent words your exit program should store registers 14 through 12. Three of the saved values have significance, as follows:

- The saved contents of register 14 contain the address within CICS to which the user exit program returns control.
- The saved contents of register 15 contain the address at which the user exit program has just been entered.
- The saved contents of register 1 address the parameter list (DFHUEPAR) that is provided by CICS for the task-related user exit program. Example:

```
USING     *,15
STM       14,12,12(13)
              .
              .
              .
USING     DFHUEPAR,1
L         2,UEPEXN
USING     DFHUERTR,2
CLI       UERTFGP,X'00'
BNE       NOTTRUE
              .
              .
              .
```

This example saves the CICS registers and then maps the CICS-provided parameter list (addressed in register 1) onto the structure DFHUEPAR. It can then load the contents of UEPEXN into register 2. This gives access to DFHUERTR, the function definition. The code then tests UERTFGP within the function definition to see if it is zero. A zero setting of UERTFGP indicates that this is a task-related user exit call. If it is not zero, the code branches to an error routine.

A macro-level assembler application program must place the address of a register save area in register 13 before invoking the stub. The application program must restore the CSA address in register 13 on return from a call to the stub. This is done automatically for macro-level COBOL and PL/I programs, and for all command-level programs.

Note: As a general rule, if you fail to understand the origin or the purpose of a call, you should:

1. Restore any registers that you have used to the state they were in on entry to your code.
2. Return to the address contained in CICS register 14.

The calling program's registers

The calling program's registers are stored at the address specified by UEPHMSA of DFHUEPAR. Where the calling program is a CICS management program, for example the syncpoint manager, the only caller registers that have significance are registers 1 and 15. Register 1 addresses the calling program's parameter list. CICS sets the calling program's register 15 to zero before the task-related user exit program is invoked. The calling program's register 15 is subsequently used to pass responses back to the calling program from the task-related user exit program. If the calling program is a CICS management program, and the register is still zero on return, CICS assumes that its call was not understood. If the calling program is an application program, the significance of register settings on return will either be described in your resource manager's documentation, or be defined locally.

Using CICS commands in your task-related user exit program

You may find some CICS commands useful in your exit program. These can be invoked using CICS command-level or macro-level statements. However, you should take note of the following:

1. If your exit program entry point is immediately followed by an occurrence of a DFHEIENT macro, inserted either implicitly by CICS or explicitly in the program, then the expansion of the DFHEIENT macro stores incorrect values at DFHEIBP and DFHEICAP. Your code can subsequently correct this by copying UEPEIB into DFHEIBP, reloading the EIB base register (DFHEIBR) from UEPEIB, and setting DFHEICAP to X'80000000'. For example,

```
TESTPROG DFHEIENT CODEREG=2,EIBREG=11,DATAREG=10
         USING DFHUEPAR,1
         MVC   DFHEIBP,UEPEIB                    Get correct EIB address
         L     DFHEIBR,UEPEIB                    Reload EIB base register
         MVC   DFHEICAP,=X'800000000'
```

Note that the entry point of a program does not have to be at the start of the program and may be positioned after the DFHEIENT macro.

2. The DFHEIENT macro allocates dynamic storage to be mapped by the DFHEISTG DSECT. In this case, you must return to CICS by means of the DFHEIRET macro, which frees the dynamic storage.
3. Command-level calls use registers 0, 1, 14, and 15.
4. If your user exit program contains macro-level calls, you must set up registers 12 and 13 to address the TCA and CSA, using the fields UEPTCA and UEPCSA in the user exit parameter list, before you issue the macro-level call.
5. The response to an EXEC CICS RETURN or EXEC CICS ABEND within your exit program is unpredictable. Rather than issue these commands, your exit program should send a bad return code to the calling program in the calling program's register 15.
6. On each invocation of a task-related user exit program, a new EXEC environment is created, even when the program is being invoked from the same task. This means that CICS operations, such as a browse of a resource definition table, cannot be continued from one invocation of the exit program to the next.

Work areas

When you use the EXEC CICS ENABLE command to identify a task-related user exit program to CICS, you may specify that the program must have access to one local and/or one global work area. The EXEC CICS ENABLE command allows you to specify the size, in bytes, of the work areas to be acquired for your task-related user exit program. CICS acquires storage for the areas and initializes pointers to them. The user exit parameter list, DFHUEPAR, gives you access to the pointers. For more information, see the description of DFHUEPAR under "User exit parameter lists" on page 348.

The global work area

A global work area is associated with an exit program. Whenever the exit program is invoked, it has access to the area through the parameter UEPGAA of DFHUEPAR. The global work area may be shared by a number of exit programs. The area can be thought of as a logical extension to the common work area (CWA, CSAWABA) that is exclusively for the exit program's use. You must have specified the size of the global work area using the GALENGTH parameter or the GAENTRYNAME parameter of the EXEC CICS ENABLE command.

The local work area

A local work area is associated with a single task and lasts only for the duration of the task. It is for the use of a single task-related user exit program. It can be thought of as a logical extension to the transaction work area (TWA, TWACOBA) that is exclusively for the exit program's use. It is specified using the TALENGTH option of the EXEC CICS ENABLE command and is accessed using the UEPTAA parameter of DFHUEPAR.

Using the CICS syncpoint manager with your task-related user exit program

All task-related user exit programs can be invoked by the CICS syncpoint manager. An exit program must 'schedule' the syncpoint manager by setting the syncpoint manager bit in the schedule flag word. The flag word must be set after every piece of recoverable work in order to ensure that the CICS syncpoint manager calls the exit program. The identification of each unit of recovery (or logical unit of work) is addressed by the eight-byte field UEPURID. This is available on all invocations of your exit program in which recoverable actions are possible, for example, application calls, and subsequent syncpoint manager calls.

Note: Be sure not to request a syncpoint from within an exit that was itself invoked by the CICS syncpoint manager. This would cause a recursion into the CICS syncpoint manager that cannot be supported. You should always avoid recursive requests.

What is expected of your resource manager

If the protocols implicit in the syncpoint manager commands are observed (that is, if every request from the syncpoint manager prompts a meaningful response from the resource manager), CICS ensures that changes to recoverable resources (such as databases) can be synchronized. That is, either all the changes will take effect or all will be backed out, even across system failures. The CICS syncpoint manager parameters are described on page 351.

On receiving 'Prepare to Commit', the resource manager is expected to get into a state where recoverable changes made since the last syncpoint can be either committed or backed out on demand, even if there is an intervening system failure. For example, buffer contents must be moved to nonvolatile storage. If the resource manager is unable to get into this state, the exit program should use a register 15 return code UERFBACK to request backout. Normally it should set register 15 to indicate a 'Yes-vote' (UERFPREP). Note that "register 15" in this section refers to the syncpoint manager's register 15, the fifth word of the area addressed by UEPHMSA.

On receiving 'Commit Unconditionally' or 'Backout', your resource manager should take the corresponding irreversible step, and have the exit program send the syncpoint manager a return code; either UERFDONE, meaning 'Done — the commit or abend process is complete', or UERFHOLD, meaning 'Not done — please remember the commit or abend for later resolution'. These return code constants are available to you when you code the macro instruction DFHUEXIT TYPE=RM in your exit program.

If a resource manager cannot understand a call, it should not change the contents of the caller's register 15 before returning to the caller, because it cannot anticipate how the caller will interpret the change.

Resynchronization after failure

Should a failure occur between returning from the 'Prepare to Commit' exit and the subsequent 'Commit Unconditionally' or 'Backout', the resource manager must be ready, on restart, to discover the state of the unit of recovery, and to act accordingly. Restart resynchronization is described on page 365.

CICS initialization and keypoint management routines recover from the system log all information associating resource managers with outstanding units of recovery. They keep the information until the outstanding units of recovery are resolved.

Using the CICS task manager with your task-related user exit program

If the task manager bit is set in the schedule flag word, your exit program will be invoked at start- and end-of-task. To determine whether a particular invocation is at start-or end-of-task, you can examine the CICS task manager parameters described in "CICS task manager parameters" on page 352. Typically, your program would show interest in task manager events if it needed to save task-related information, such as performance or accounting data, before the task ended. If you use such an exit, you should respond by passing a return code (for example, UERFEOTR "call was understood") to task management in register 15. Return code constants, such as UERFEOTR, are available to you when you code the macro instruction DFHUEXIT TYPE=RM in your exit program.

If your exit program is invoked at end-of-task, you must be alert to possible limitations on exit program activity at task-detach. For example:

- Do not update a recoverable CICS resource during a task-detach exit call because the CICS syncpoint manager will not be invoked again for that task. Note also that all resources (terminals, and so on) except task-storage have been released by end-of-task.
- It is possible to schedule a new CICS task from your exit program using the EXEC CICS START command (or its macro-level equivalent), and to pass data to a new task. However, you should note that EXEC CICS START uses a temporary storage queue to pass data to the new transaction. If this queue is recoverable (DFHTST TYPE=RECOVERY), it will be locked to the detaching task. It will never be unlocked, because when the task-detach exit call is made, the resources of the detaching task have **already** been freed. Use of the PROTECT option would cause a different problem: the new task could not be scheduled until the next syncpoint of the detaching task, but there will be no such syncpoint.

We recommend that you do not access remote resources using a task-related user exit program. However, if you do so, you must understand fully the circumstances in which the function-shipping conversation may be terminated.

Adapter administration

Careful use of task-related user exits can allow your application programmers to be unaffected by the invocation of non-CICS resource managers from CICS application programs. Enabling and disabling task-related user exit programs for an installation should be the responsibility of one or more supervisory or master terminal operators. This section lists what you must do before you can use the adapter, and describes the commands used by the supervisor to administer task-related user exit programs.

The general rules about the use of commands in CICS application programs are given in the *CICS/MVS Application Programmer's Reference* manual.

What you must do before using the adapter

1. Ensure that your CICS system was initialized with EXITS=YES in DFHSIT.
2. A task-related user exit program must be defined to the system. To do this you may either use the CEDA transaction, or create a PPT entry using macro-level instructions.
3. If you want to use CICS syncpoint management in task-related user exits, your CICS system must contain the supplied module DFHDBP.
4. To enable the task-related user exit program and to define its working storage needs you must use the EXEC CICS ENABLE command. A task-related user exit program must be both ENABLEd and STARTed before it is available for execution. The commands are documented in this section, starting on page 361.

Tracing a task-related user exit program

CICS will output a trace entry just before control is passed to the task-related user exit and just after returning from the exit. These trace entries may be controlled by using the EI option of the EXEC CICS TRACE ON and OFF command. They may also be controlled by specifying the EI option on the STYPE operand of the DFHTR macro in macro-level application programs.

Installing and withdrawing exit programs

You use the EXEC CICS ENABLE and DISABLE commands to install and withdraw task-related user exit programs. The commands are similar to those used for global user exits.

You should prepare procedures for enabling and disabling your task-related user exit programs, and for resynchronizing between sessions or after a system failure. For an explanation of resynchronization, see "Restart resynchronization" on page 365.

Often you will want an exit program to be enabled at the start of a CICS run. You can code a program containing an EXEC CICS ENABLE command, followed by a RESYNC command to ensure that no in-doubts (that is, incomplete units of recovery) remain from the previous session. You can define a transaction to invoke the program, and restrict its use to particular operators.

If you want to enable a particular task-related user exit program at the start of every CICS run (that is, the resource manager is always part of your system), you can create an entry for this enabling program in a program list table invoked during CICS postinitialization.

Note: The enabling and disabling of an exit program overrides, but does not alter, the enable or disable status for the program in the processing program table (PPT) entry or in the CICS system definition file.

One load-module can contain several task-related user exit programs. If a load-module contains more than one exit program, each program has one entry point that must be named using the ENTRYNAME parameter of the EXEC CICS ENABLE command. If you do not specify an ENTRYNAME, the default value will be taken from the name specified on the PROGRAM parameter, which is the name of the load-module itself.

Enabling an exit program

A task-related user exit program is enabled in two stages, as follows:

1. Load the task-related user exit program and obtain work areas.
2. Make the task-related user exit program available for execution.

The two stages are performed by using two EXEC CICS ENABLE commands, as illustrated on page 363.

The EXEC CICS ENABLE command

This command identifies a named task-related user exit program to CICS.

```
EXEC CICS ENABLE PROGRAM(name)
                 [START]
                 [ENTRY(pointer-value)]
                 [ENTRYNAME(name)]
                 [GALENGTH(data-value)|
                    GAENTRYNAME(name)]
                 [TALENGTH(data-value)]
                 [TASKSTART]
```

PROGRAM(name)
Specifies the name of the **load module** of the exit program. The name can be any character string up to eight bytes, and it must be the name of a program in the PPT or in the CICS system definition file.

START
Specifies that the task-related user exit program is to be made available for execution.

ENTRY(pointer-value)
Specifies the entry address of the task-related user exit program. If this operand is specified, CICS assumes that the exit program is already loaded and will not attempt to load it, nor will it attempt to delete it when the exit program is disabled, even if the EXITALL option is specified on the EXEC CICS DISABLE command. The specified address must be within the virtual

storage range occupied by the exit program. If this operand is not specified, the exit program is loaded by CICS, the entry address returned from the load is used, and CICS will delete the exit program when it is disabled.

ENTRYNAME(name)
Specifies the name of this entry to the task-related user exit program. This name need not be defined in the CSD or PPT. It must be unique among enabled entry names. If omitted, the value will be taken from the mandatory PROGRAM argument. Its presence **does not** require the ENTRY keyword to be specified.

Note: The same combination of ENTRYNAME/PROGRAM that is specified on the initial EXEC CICS ENABLE command must be used on subsequent EXEC CICS ENABLE, DISABLE, and EXTRACT EXIT commands directed to the named entry.

GALENGTH(data-value)
Specifies the length, in bytes, of the global work area that is to be provided by CICS for this exit program. If a data variable is specified, it must represent a halfword binary data item. Valid lengths are 1 through 32767. The work area will be initialized to binary zeros.

GAENTRYNAME(name)
Specifies the name of a currently enabled task-related user exit program whose global work area is also to be used by the exit program being enabled. The exit program specified must own the work area (that is, GALENGTH must have been specified when the named exit program was enabled). If a work area is shared by two or more exit programs, it is not released until all these exit programs are disabled. However, after the owning exit program is disabled, no new exit program can share the work area.

GALENGTH and GAENTRYNAME are mutually exclusive. If both operands are omitted, no global work area is provided.

TALENGTH(data-value)
Specifies the length, in bytes, of the local (task-related) work area that is to be provided by CICS for this task's invocation of the exit program. This work area is released at the end of the task for which it was created. "Data-value" can identify a variable, which in turn represents the length. Valid lengths are 1 through 32767. CICS initializes the work area to binary zeros before first giving control to the task-related user exit program. If you do not specify TALENGTH, CICS does not create a local work area.

TASKSTART
Enables your task-related user exit program to be invoked automatically at the start and end of every task. (There is an exception to this rule that arises when logging off an autoinstalled terminal in an MRO environment. The module that processes the 'remote delete' does not return control to the CICS module that handles task-related user exits, and therefore the TASKSTART invocation cannot be made.) If you specify TASKSTART, application programs can still invoke the exit program. This option is independent of the START option above, but you should also specify START, unless a previous EXEC CICS ENABLE command (which specified START but not TASKSTART) has already made the exit program available for execution.

See "User exits for accessing monitoring data" on page 416 for a typical use of this option.

On the second and subsequent EXEC CICS ENABLE commands for a particular exit program, ENTRY, GAENTRYNAME, GALENGTH, and TALENGTH must not be specified and either START, or TASKSTART, or both must be specified.

Examples: The enable-start sequence for an exit program can be done using two EXEC CICS ENABLE commands.

1.

```
EXEC CICS ENABLE PROGRAM('EP9')
   TALENGTH(750) ENTRYNAME('RM1') GALENGTH(200)
```

2.

```
EXEC CICS ENABLE PROGRAM('EP9')
   ENTRYNAME('RM1') START
```

The first command loads the task-related user exit program EP9, and causes a 200-byte work area to be obtained and associated with the name RM1. It also schedules the allocation of a further 750-byte work area for each task that subsequently invokes RM1. The second command starts the exit program, that is, it makes its entry point capable of being invoked.

Disabling an exit program

You use the EXEC CICS DISABLE command to disconnect a task-related user exit program from CICS. When the program has been disabled, CICS will not route application program requests to the resource manager.

Generally, an exit should be disabled using the EXEC CICS DISABLE command with the EXITALL option. This causes CICS to delete the exit program, and release all work areas associated with it. Should it be necessary, you can make the exit program unavailable for execution without deleting either the program or its global work areas. To do this, you code EXEC CICS DISABLE with the STOP option (not EXITALL).

The EXEC CICS DISABLE command

This command disables a named user exit program, thus preventing CICS from handling calls to the exit program. The format of the command is:

```
EXEC CICS DISABLE PROGRAM(name)
                  [ENTRYNAME(name)]
                  [EXITALL]
                  [STOP]
                  [TASKSTART]
```

PROGRAM(name)
Specifies the name of the load-module of the task-related user exit program.

ENTRYNAME(name)
Specifies an enabled entry name. Its default is taken from the PROGRAM argument. For successful execution of the EXEC CICS DISABLE command, the same combination of ENTRYNAME/PROGRAM must be used as was specified in the original EXEC CICS ENABLE.

EXITALL
Specifies that the exit program is to be deleted from main storage. You should avoid requesting this function when applications that have used the task-related user exit are still running (if you do, the results are unpredictable). EXITALL implies STOP.

STOP
Has the reverse effect of the START option on the EXEC CICS ENABLE command. It specifies that the task-related user exit program is to be made unavailable to calling programs, though it is to remain in main storage. This function can better be performed by your exit program responding to calls from your API or from the syncpoint manager, for example, on a task-by-task basis. You are therefore advised not to use the STOP option. The EXITALL option should serve your purposes.

TASKSTART
Allows you to dissociate your exit program from the start-of-task exit without fully disabling the exit program.

At least one of the operands TASKSTART, EXITALL, and STOP must be specified. TASKSTART and EXITALL are mutually exclusive. If both are omitted, the exit program remains in virtual storage.

Examples: The stop-disable sequence will usually be performed in a single EXEC CICS DISABLE command. The following command deallocates and conditionally deletes the exit program RM1.

```
EXEC CICS DISABLE PROGRAM('EP9') ENTRYNAME('RM1') EXITALL
```

CICS does not prevent this type of DISABLE when control is in a task-related user exit. Tasks connected to the task-related user exit will abend when they next communicate with it.

An INVEXITREQ condition with EIBRCODE bytes 2 and 3 equal to X'0080' is possible for a task-related user exit, but only during a brief period just **before** its entry-point is invoked.

The EXTRACT EXIT command

You use this command to obtain information about the global work area of an enabled and started task-related user exit program.

```
EXEC CICS EXTRACT EXIT PROGRAM(name)
                       [ENTRYNAME(name)]
                       GASET(pointer-ref)
                       GALENGTH(data-area)
```

EXIT
Specifies the type of control block.

PROGRAM(name)
Specifies the name of the load-module of the task-related user exit program.

ENTRYNAME(name)
Specifies the name of an enabled entry. The address and length of the global work area associated with this entry name are to be extracted from the control block. The exit program can either own or share the work area. By default it takes the value specified by the PROGRAM argument. The same combination of ENTRYNAME/PROGRAM must be specified as on the original EXEC CICS ENABLE command.

GASET(pointer-ref)
Specifies the variable that is to be set to the address of the global work area used by the task-related user exit program.

GALENGTH(data-area)
Specifies the variable that is to be set to the length of the global work area used by the task-related user exit program. It must be a halfword binary data item.

Restart resynchronization

The RESYNC command prompts the CICS syncpoint manager to diagnose any inconsistency on restart between CICS and a non-CICS resource manager.

You should issue this command from an administration routine (or from the task-related user exit program) to identify any 'in-doubts' (that is, any logical units of work,or LUWs, which may not have been committed or backed out during the previous session), and to allow CICS to purge obsolete storage of LUWs that prove not to be in doubt. This should be done after issuing an EXEC CICS ENABLE command.

If there are actual in-doubts, the RESYNC command will complete the syncpointing procedure that was interrupted for each LUW. That is, there will be one RESYNC task for each in-doubt. Your task-related user exit program does not need to include special routines for resynchronization. The RESYNC mechanism looks like any other syncpoint call to your exit program.

A description of the parameter values that the CICS syncpoint manager provides after the issue of an EXEC CICS RESYNC command is given on page 351.

It is recommended that you issue the ENABLE-with-START command before issuing a RESYNC command. If you delay the ENABLE-with-START command until after the RESYNC command, ensure that you do not lose control between the RESYNC command and the ENABLE-with-START command. If you do lose control, the RESYNC-task may encounter your not-yet-started exit program and will treat this as a failing situation, making provision for another RESYNC attempt in the future.

The format of the command is:

```
EXEC CICS RESYNC ENTRYNAME(ename)
                 [IDLIST(data-area)]
                 [IDLISTLENGTH(data-area)]
```

The options have the following meanings:

ENTRYNAME
Specifies the name of the entry point of the task-related user exit program. It should be the same as the name specified both in the TO operand of DFHRMCAL, and in the EXEC CICS ENABLE and DISABLE commands.

IDLIST
Specifies an address-list provided by the resource manager, the last address of which is identified by a X'80' bit in its high-order position. The addresses in the list locate CICS unit of recovery identifiers.

Each identifier represents an in-doubt unit of recovery, and will have been presented originally to the exit program (during a previous lifetime) as the argument of the UEPURID parameter of the user exit parameter list.

IDLISTLENGTH
Specifies a halfword containing the length (in bytes, counting four bytes per in-doubt) of the address-list. The IDLISTLENGTH parameter must be coded if the IDLIST parameter is specified.

Note: If your adapter is designed to resynchronize following system failures, you are recommended always to issue the RESYNC command, even during a start-up in which your resource manager has no in-doubts. This allows CICS to avoid accumulating storage of **possible** in-doubts. This is achieved either by use of a RESYNC command without the IDLIST and IDLISTLENGTH parameters, or by having IDLISTLENGTH specify a halfword of binary zero.

EIB function codes

The following are the function codes of the task-related user exit commands.

EIBFN Code	Command
22 02	EXEC CICS ENABLE
22 04	EXEC CICS DISABLE
22 06	EXEC CICS EXTRACT EXIT
16 04	EXEC CICS RESYNC

Exceptional conditions

All errors in the ENABLE, DISABLE, and EXTRACT EXIT commands are grouped under the single INVEXITREQ exceptional condition. The default system action for the INVEXITREQ exceptional condition is to terminate the transaction with abend code AEY0. There are no exceptional conditions associated with the RESYNC command.

For details of how to code tests for the INVEXITREQ condition using DFHRESP, see the *CICS/MVS Application Programmer's Reference* manual.

If byte 0 of EIBRCODE is X'80', an error has occurred. The exact cause of the error can be determined by examining bytes 1 and 2 of EIBRCODE, which can have the following hexadecimal values:

EIBRCODE bytes 1-2	Command	Meaning
X'8000'	ENABLE	The load module named in the PROGRAM option is AMODE=31, or is not in the PPT or the load library, or its PPT entry has been disabled.
X'2000'	ENABLE	Program specified in ENTRYNAME option (or defaulted to PROGRAM argument) is already enabled.
X'0800'	ENABLE	Exit program specified in GAENTRYNAME option is not enabled.
X'0400'	ENABLE	Exit program specified in GAENTRYNAME option does not own a work area.
X'0040'	ENABLE	CICS system was not initialized with EXITS=YES in DFHSIT.
X'8000'	DISABLE	The load-module named in the PROGRAM option is not in the PPT, or is not in the load library, or its PPT entry has been disabled.
X'0200'	DISABLE	Program is not enabled.
X'0080'	DISABLE	Program is currently invoked by another task (see note).
X'0040'	DISABLE	CICS system was not initialized with EXITS=YES in DFHSIT.
X'8000'	EXTRACT EXIT	The load-module named in the PROGRAM option is not in the PPT, or is not in the load library, or its PPT entry has been disabled.
X'0400'	EXTRACT EXIT	Program has no work area.
X'0200'	EXTRACT EXIT	Program is not enabled.
X'0040'	EXTRACT EXIT	CICS system was not initialized with EXITS=YES in DFHSIT.

Note: The INVEXITREQ condition with X'0080' in bytes 1 and 2 can occur only if a task switch has occurred in the exit program due to a request for a CICS service. The normal action for this condition is to retry the DISABLE request. However, if such an exit program terminates abnormally, its use count remains greater than zero and it cannot be deleted, but it can be stopped.

End of General-Use Programming Interface

Chapter 5.5. Writing postinitialization and termination programs

Postinitialization programs

When writing programs to be executed during the postinitialization phase, you should consider the following factors:

- All CICS facilities except direct communication with terminals are available to PLTPI (program list table post-initialization) programs. Because interregion communication (IRC) and intersystem communication (ISC) have pseudoterminal entries associated with their function, you cannot run any IRC or ISC functions during PLT processing at system initialization time. This includes performing inquiries on these IRC/ISC functions.
- User programs to be executed during postinitialization must be listed by name in a PLT, and the suffix of that PLT must be specified in the PLTPI system initialization parameter. Programs that are initiated from the PLTPI list have no access to a TWA.
- PLTPI programs must not request any service that could logically suspend or abend the task. Suspending or abending the TCA of the terminal control program will cause the system to be terminated abnormally. PLTPI programs must not depend on transactions being initiated by interval control, because no interval control initiations occur until the postinitialization phase has completed.
- A PLTPI program must not explicitly issue an abend. Also, you must be aware of implied abends that may be issued on your behalf if a HANDLE CONDITION has not been coded.
- A PLTPI program must not issue a dump request.
- A PLTPI program must never change the priority of the task executing the PLTPI program, because the task is the TCP and it must remain as the highest priority task.
- Although other tasks can be attached during PLTPI processing, they must not access a protected resource that is also being accessed by the TCP task that is executing PLT programs.

  You can use task control ENQ/DEQ facilities in the PLT programs executed by the TCP task and in other programs executed by the attached programs to ensure single threading of the use of protected resources. However, it is your responsibility to ensure that the TCP task always enqueues first. (If TCP attempts an ENQ and fails to gain control of the resource, it will be suspended, and will cause CICS to abend.)

  Because concurrent accessing of protected resources is difficult to control, it is recommended that the access be serialized, and that no task is attached to access the resource until the TCP task executing the PLTPI programs has completed its processing.
- Because standard CICS services are available to PLTPI programs, it is important to understand and consider the effect of these services when they involve accessing protected resources as defined to CICS.

When a protected resource is accessed, CICS normally enqueues on the resource to ensure exclusive ownership, during the task's use of the resource. The dequeuing of a protected resource is deferred until the task terminates or voluntarily declares itself to be at a sync point (through the EXEC CICS SYNCPOINT command). However, the use of a syncpoint rollback should be avoided in a PLT program.

PLTPI programs that are involved in rebuilding a protected file will cause an enqueue to occur for each logical record they access. Because the dequeues are deferred, it is advisable for you to declare syncpoints throughout the recovery process to allow dequeues to occur. If this procedure is not followed, dynamic storage can become filled with CICS control blocks used to control the enqueue/dequeue facility.

In addition, no other task should be attached that could also access records in the database until after the PLTPI program has completed its entire rebuilding operation. Enqueuing on the database by both tasks will not keep the attached task from gaining control of the database when the PLT processing task declares a syncpoint. This is because of the implicit dequeuing that occurs at that time.

- PLTPI programs that invoke the use of the programmable interfaces (DFHEMTA, DFHEMTP) to use the CEMT functions can cause the table management function to be invoked. Table management enqueues on resources by establishing lock blocks to ensure exclusive ownership for the duration of the task, in this case TCP's task. These blocks will normally only be released at the end of the task.

Shutdown (PLTSD) programs

When writing programs to be executed during system shutdown, remember that the PLTSD programs run either in the first quiesce stage, or in the second quiesce stage:

- Terminals are still available during the first quiesce stage, but tasks started by terminal input will be rejected (with a few exceptions) unless they are named in a shutdown transaction list table (XLT). The exceptions are the CICS service transactions (CSMT, CEMT, CSAC, CSIR, CSTE, CSNE, and CSSF), which are always accepted. are named in an XLT. Tasks started by other means, for example, automatic transaction initiation (ATI), will be allowed to proceed.

  The first quiesce stage is considered to be complete when first-stage PLT programs have been executed, and when there are no user tasks in the system.

- The second quiesce stage begins at the point in the PLTSD table defined by DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM. Termination activity waits until all first-quiesce-stage system activity stops. Termination then continues with the TCP and task control ATTACH disabled; when all PLTSD programs defined to be executed in the second quiesce stage have been completed, CICS stops.

Note that programs specified for the second quiesce stage are not allowed to use any terminal control services or task control ATTACH requests. No automatically initiated transaction can be initiated during the second quiesce stage.

- If your PLTSD program abends, syncpoint rollback will occur.
- If a transaction abend occurs while the PLTSD program is running, CICS is left in a permanent WAIT. To avoid this happening, ensure that your PLTSD program handles *all* abend conditions.
- PLTSD programs will not be executed during an XRF takeover by the alternate.

Chapter 5.6. System initialization overlays

This chapter contains information on the conventions and general rules that must be observed when writing CICS system initialization overlays. Before attempting to add an overlay to system initialization processing, you should have a thorough knowledge of CICS internals, be proficient in assembler-language coding, and have a reasonable knowledge of the operating system you are using.

User-written overlays may be added to the system initialization program; however, you are warned that the interface to user-written overlays is subject to change in later releases of CICS.

Overlays must conform to CICS naming conventions. All system initialization overlays are seven-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 (for example, A1, B1,...Z1). User overlays may use any other two-character suffix.

Overlay processing in system initialization is driven from the system initialization table SIMODS parameter. User-written overlays may be inserted at any point in system initialization processing, but the sequence of CICS overlays must not be disturbed.

Because of the complexity of CICS, various operating system considerations, and your needs, it is impossible to describe here all the considerations for coding user overlays. CICS is responsible for common subroutine and overlay linkage (assuming these routines are not modified), and normal system initialization functions.

General rules for overlay coding

Here are some general rules for overlay coding.

- All overlays must be coded in assembler language.
- All overlays must include the DFHSICOM macro (system initialization common area); this provides all system initialization register definitions, commonly used equates, and DSECTs, for the system initialization communications area, DFHSIPDS, the system initialization table, DFHSIT, the processing program table, DFHPPT, and the common system area. Other DSECTs must be included as required.

  Note that in some cases the registers are used to pass information between CICS system initialization modules. This information might be lost if the user-written system initialization module makes use of these registers for other purposes. Therefore, you must take care when using the registers, otherwise unpredictable abends might occur.
- All overlays must contain the following two USING statements immediately prior to the first executable instruction.

```
USING *,SIPBAR2[,SIPBAR3]       Establish program addressability
USING SIPCOM,SIPBAR1            For common area addressability
```

- All overlays must exit through the system initialization overlay supervisor.
- Overlays must not exceed 8,192 bytes.
- User-written system initialization overlays must communicate with each other using shared storage rather than task-local storage.

System initialization subroutines

Here is a list of system initialization subroutines and conventions for calling them:

1. SIPCORE — common storage allocation subroutine for overlays executed before DFHSIB1.

   Calling sequence:

   ```
   L       SICORA,=F'500'              Load storage required
   L       SILINKR,SIPCORE             Get allocation routine address
   BALR    SILINKR,SILINKR             Go get storage
   ```

   Return sequence:

   Symbolic register SICORA contains the address of acquired storage. All other registers are unchanged.

   Note: This routine is not available between DFHSIB1 and DFHSIH1, and so you should use GETMAIN.

   After DFHSII1, DFHSC TYPE=GETMAIN should be used.

2. SIPBLDL — common BLDL subroutine

   Calling sequence:

   ```
   MVI     SIPFLAG,SIPBLNUC            BLDL for nucleus module
   MVC     SILISTID,=CL8'routine name' Move name
   L       SILINKR,SIPBLDL             Get routine address
   BALR    SILINKR,SILINKR             Go to routine
   ```

   Return sequence:

   SIPARMP3 Contains storage required for load module

   SILISTTR Contains TTRK for load module

   All registers except register 15 are unchanged.

3. SIPLDER — system initialization program loader

   Calling sequence:

   Symbolic register SICORA contains storage address to load program.

   SILISTTR — Contains TTRK of load module

   ```
   MVI     SIPFLAG,SIPBLNUC        Load for nucleus module
   L       SICORA,=A(load point)   Point at place to load
   MVC     SILISTTR,ttrk           Move TTRK
   L       SILINKR,SIPLDER         Get loader address
   BALR    SILINKR,SILINKR         Go load module
   ```

   Return sequence:

   Symbolic register SICORA points at load point of program. All other registers are unchanged.

4. SIPOSUP — system initialization overlay supervisor

   Calling sequence:

   ```
   L       SILINKR,SIPOSUP         Get overlay supervisor
   BALR    SILINKR,SILINKR         Go exit
   ```

   Return sequence:

   None. Transfer is given to the next overlay of SIP.

Addresses and fields

The following areas are always addressable to system initialization overlays at entry, and must be addressable at exit.

- CSA — Common system area
- SIT — System initialization table
- SIPCOM -- System initialization common area
- System initialization common routines.

After DFHSII1, register 12 addresses the user part of the TCA of the system initialization task and must not be corrupted.

The following fields are supplied as parameter-passing fields between user overlays of system initialization. These fields are not to be used by CICS overlays.

```
SIPARMP6    FULLWORD
SIPARMP7    FULLWORD
```

The DFHWTO macro instruction is provided for use within system initialization for conditional write-to-operator (WTO) functions. If the MSGLVL in the system initialization table is 1, all messages are written; if the MSGLVL is 0, none are written. Any messages not to be suppressed should be written by means of the WTO macro instructions. The format for DFHWTO is:

```
DFHWTO          'MESSAGE UP TO 132 CHAR'
```

Chapter 5.7. CICS security management

CICS provides an interface to an external security manager that may be user-written or may be the Resource Access Control Facility (RACF) program product.

To use your own security manager, you may need to replace the supplied DFHXSP with your own version of the module.

You can replace the RACF interface module by one of your own, using the system modification program (SMP). Put your own version in place of the version of DFHXSE supplied in CICS.LOADLIB.

If you replace DFHXSP or DFHXSE, be aware that you are replacing part of the CICS nucleus. The following restrictions apply:

- You must use the CICS macro-level interface (**do not use any EXEC CICS commands**).
- You should save any control block data that you need and restore it when your replacement modules have finished. This is because CICS macros can change the contents of CICS control block fields.

Security modules

This part of CICS performs all security checks and signon verification. It consists of **two modules**: DFHXSP and DFHXSE.

- DFHXSE is used to pass requests to an ESM (external security manager).
- DFHXSP is the main module. It performs the CICS security checking and the search of the signon table.

Under MVS the RACF macros are invoked through the CICS SVC and the SVC phase DFHXSS. DFHXSS uses register 1 to pass the address of a parameter list, which is the common interface between DFHXSP, DFHXSS and DFHXSE. You can address this parameter list using the DFHSEC TYPE=DSECT macro instruction (see Figure 29 on page 379).

Note: This parameter area has been modified from that specified in earlier releases. The SNTTE now contains all the terminal related security data. If a check call is passed the address of the TCTTE, then the SNTTE may be found from the field TCTTESNT.

The module DFHXSP is called at transaction attach, initialization, signon, sign-off, and to perform minor functions such as time-out. The possible request codes in the second byte of the parameter area (see Figure 29 on page 379) identify the type of call, as follows:

The initialization call (DFHSII1) request code 0
Is made once in order to attach a CICS task that will prime the resource class blocks (descriptors of the resource), and for RACF to build the resource profiles. You can use the field CSASECBL to access the class blocks. The DSECT XSPSCBDS describes the order of the class blocks.

The signon with password call (DFHSNP) request code 4
Is made in order to compare the userid and password entered with those held in the CICS signon table, or to request RACF to verify the userid. The SECFMT3 indicator shows whether the SECUID field contains a pointer to a 20-character operator name or an eight-character userid. The security values from the signon table entry and the pointer to the RACF control block, ACEE, are stored in a pseudo signon table entry, SNTTE. The address of the SNTTE is returned in register 0. Note that DFHSNP stores the address of the SNTTE in the TCTTE (TCTTESNT).

The signon without password call (DFHZNAC) request code 8
Is made to establish authority for links in IRC/ISC connections and to build the SNTTE.

The resource check call (DFHZSUP,DFHEIP) request code 12
Is made to determine whether the signed-on user may access the specified CICS resource, or is made to attach a transaction. The SNTTE contains the security related data. The address of the CICS resource (PPT, for example) is passed at offset 8 (SECRSNM) in the parameter list. The properties of the resource are defined in the resource class block.

The sign-off call (DFHSFP) request code 16
Is made to free the SNTTE block and to sign off the user from RACF.

The sign-off call after timeout (DFHCSSC) request code 20
Will free the SNTTE block and sign off the user from RACF, provided that SIGNOFF=YES has been coded in the TCT.

The return USERID call (DFHEEI) request code 24
Is made to pass back in register 0 a pointer to the length of the USERID followed by its value in the SNTTE that was built at signon time.

The wait for initialization call (DFHSII1) request code 28
Is made to synchronize the initialization tasks for the various management functions.

The return minimum time-out call (DFHCSSC) request code 32
Is made to return in register 0 the minimum time-out value from the signon table (SNT).

The build pseudo-SNTTE call (RDO) request code 36
Is made when installing a TCT entry with preset security.

The delete pseudo-SNTTE call (RDO) request code 40
Is made by RDO when removing a TCT entry with preset security.

The rebuild call (CEMT) request code 44
Is made to refresh the RACF resource profiles.

The internal initialization call (DFHXSP) request code 48
Is made by DFHXSP in response to an initialization call and is part of the CICS initialize security task process.

0	Access flags	Request code	Format	Reserved
4	Address of userid			
8	Address of resource block / address of RSLKEY			
C	Address of old password / address of OPIDENT			
10	Address of new password / address of SCTYKEY			
14	Address of operator identification card text / address of OPPRTY			
18	Address of TCTTE			
1C	Address of resource class block / address of operator class			
20	Address of pointer to pseudo SNTTE			
24	Reserved			

Figure 29. Parameter area (DSECT DFHSECDS) for DFHXSP

In Figure 30, X indicates that the corresponding field may be passed in the DFHXSP/DFHXSE parameter list. The SNTTE block should be referenced to access any security data on a check call. Not all the indicated fields are necessarily passed.

DFHXSP parameter fields	Request code													Offset in parameter area
	0	4	8	12	16	20	24	28	32	36	40	44	48	
SECUID		X	X											4
SECRSNM				X										8
SECOPSW		X												C
SECNPSN		X												10
SECOPID		X												14
SECTCTTE		X	X	X	X	X	X			X	X			18
SECCLASS				X										1C
SECSNTTE				X	X	X	X							20
SECRSLKY										X				8
SECOPIDT										X				C
SECSCTY										X				10
SECOPPRY										X				14
SECOPCLS										X				1C

Figure 30. DFHXSP/DFHXSE parameter list

Note: If both TCTTE and SNTTE are passed, the SNTTE should be referenced from the TCTTE.

All character strings addressed by the parameter list, except the user identification name when connecting systems, and the OPIDENT field, consist of a one-byte length field followed by the text string. The format of the resource class block is shown in Figure 31 on page 380.

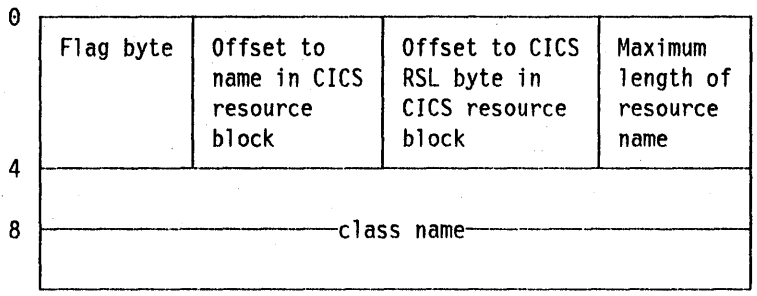

0
Flag byte
Offset to name in CICS resource block
Offset to CICS RSL byte in CICS resource block
Maximum length of resource name
4
8
class name

Figure 31. Format of the resource class block. See the CICS/MVS Data Areas manual for the DSECT (XSPSCBDS).

On return, DFHXSP should place one of the following codes in register 15:

0	Successful execution
4	New password required
8	Operator identification card required
C	Reserved
10	Invalid signon attempt
14	Invalid access to a protected resource
18	External security failure or invalid parameter list.
1C	No signon table found
20	Reserved
24	Invalid sign-off attempt.

In addition, register 0 may contain a user code in the range 0 to 99. If the external security manager is RACF, this code will be the RACF return code. If an error occurs, this code will be inserted in the error message. If RACF is used as the security manager, the userid must not be more than eight characters.

If the X'80' bit is set in register 15, there is an error in the RACF router, and the return code from the router is in the rightmost seven bits of register 15 (that is, ignoring the X'80' bit).

RACF interface module — DFHXSE

This module is branched to from DFHXSP and is used to interface to RACF. You call this module for resource checking, for signon by an ESM, and so on. The resource name may be prefixed with the security name in the CSA optional features list (CSAXSNM), if you have specified EXTSEC=(,PREFIX). You set this value from the keyword USER=xxx on the MVS job card. See the *CICS/MVS Operations Guide* for sample RACF definitions and for the RACF resource class names. You can also use this module for signon, sign-off, and at initialization. When you use RACF, the CICS SVC is invoked from within DFHXSE to perform these functions.

If any CICS services are required by DFHXSE, the programmer must protect the TCA fields. The supplied save area could be used for this and, if more save area is required, a GETMAIN must be issued.

The CICS-provided interface to RACF at signon time validates that the specified operator is authorized to use the terminal and the CICS system. SMF logging of successful signons requires the appropriate level of RACF.

The additional RACF APPL validation is performed using the value of the APPLID parameter on the DFHSIT TYPE=INITIAL macro or the override of this value given at the CICS bringup. If neither of these is specified explicitly, the default APPLID of 'DBDCCICS' is used.

The terminal name that CICS will use when calling RACF for signon validation (RACINIT) may be either of the following:

- For VTAM terminals (non-surrogate TCTTE only) the value is the eight-character NETNAME from the DFHTCT TYPE=TERMINAL macro.
- For all other terminals the value is the four-character TRMIDNT from the DFHTCT TYPE=TERMINAL macro, padded with four blanks.

The RACF tables must be updated to use these names if terminal or CICS system usage is to be restricted in your installation.

If DFHXSE only is to be replaced, it must analyze the value of the request code in the parameter list to determine the type of request. The values and their meanings, which are the same as for DFHXSP, are as follows:

0	Initial call
4	Sign-on with password
8	Sign-on without password
12	Resource check
16	Sign-off.
20	Sign-off after time-out.

In order to invoke external security from DFHXSP the following bits must be set on in the associated part of the security class block (XSPSCBDS):

```
XSPSCEXR EQU (X'02') EXTERNAL SECURITY CHECKING REQUIRED
XSPSCEXA EQU (X'01') EXTERNAL SECURITY PROFILE BUILT
```

DFHXSE is a separate module called by DFHXSP. It must establish its own addressability and addressability to any areas it requires. It must save registers 3 to 13. On entry to DFHXSE, register 1 points to the parameter list (described in Figure 29 on page 379) and register 2 addresses an 18-word save area followed by a 96-byte work area. Register 12 points to the TCA and register 13 to the CSA. Register 15 contains the entry point address, and register 14 contains the return address within DFHXSP to which control must be returned. Appropriate return codes, as described for DFHXSP, should be set in registers 0 and 15, and the module should finish by branching to the contents of register 14. Note that this is a new interface for ESM calls.

Security identification module — DFHACEE

At initialization, and within module DFHIRP for MRO connections, the module DFHACEE is invoked. Its purpose is to return the security name of the system which is then stored in the CSA optional features list for use in building the resource names for security checking. At MRO-connection time, the security name is passed to the other system for comparison with the value specified for the XSNAME parameter in the TCT SYSTEM entry and it is used to build a security block for the link. If the values are the same, the connection is allowed; if there is no value for the XSNAME parameter, however, any connection is allowed. If the security names of the two systems are the same, requests transmitted to the second system will not have security checks made against them. For an ISC connection, if you do not specify an XSNAME, any check for a remotely accessed resource will fail. For an IRC connection, if module DFHACEE returns a value of 0, the security name of the system is said to be unknown, and any security checks will fail. The CICS version of module DFHACEE returns either a name of 0 or the USER name from the job card.

You can, therefore, control the degree of security of connections by writing your own version of DFHACEE, using the system modification program (SMP). You will find DFHACEE in library CICS212.SOURCE.

On entry to DFHACEE, only register 14 is set with the return address. Registers 0, 1, 14, and 15 may be used, and no other registers should be corrupted. The security name should be loaded into registers 0 and 15, and return should be made using register 14.

Transaction security

The security value against which a transaction is checked when it is attached to a TCTTE is established either:

- From the OPERSECURITY option of the CEDA DEFINE TERMINAL command or from the OPERSEC operand of DFHTCT.

  OR

- By running the transactions CESN or CSSN. This transaction copies the signon table security-values for the associated USERID into a pseudo signon table (SNTTE) entry. The TCTTE (TCTTESNT) points to this entry.

You may create the security values in the SNTTE from one source only. Use the TCT definition, or run the signon transaction.

The following table shows a summary of the information about transaction-attach security and resource security.

	Transaction-attach security		Resource security	
When is it active?	Always		When RSLC=YES is specified for the transaction	
User authorizations (keys)	OPERSECURITY OPERSEC ...	up to 64	OPERRSL SCTYKEY	up to 24
Security values (locks)	TRANSEC	1	RSL	1

Figure 32. CICS security summary

Figure 33 shows the relation between DFHTCT operands, resource definition online options, DFHSNT (signon table) operands, and the SNTTE DSECT.

Length in Bytes of SNTTE Entry	DFHTCT Macro Operands	RDO Options	DFHSNT Macro Operands	SNTTE DSECT
3	OPERID	OPERID	OPIDENT	SNNTID
1	OPERPRI	OPERPRIORITY	OPPRTY	SNNTOP
3	OPERRSL	OPERRSL	RSLKEY	SNNTASK
8	OPERSEC	OPERSECURITY	SCTYKEY	SNNTSK (3 bytes) SNNTSKE (5 bytes)
8	USERID	USERID	USERID	SNNTUSID
3			OPCLASS	SNNTOC

Figure 33. Security keywords

Notes:

1. For more information about the entries in this table, see the *CICS/MVS Resource Definition (Macro)* manual and *CICS/MVS Resource Definition (Online)* manual, and *CICS/MVS Data Areas* manual.

2. The default security for transactions is CICS security checking and all transactions will run with a security key of 1.

Resource security

Once a command-level programming transaction is attached, it may need resource security-level checking. You specify this with the RSLC(YES) option of the CEDA DEFINE TRANSACTION or CEDA ALTER TRANSACTION commands, or the RSLC=YES operand of the DFHPCT TYPE=ENTRY macro. The CEDA transaction is described in the *CICS/MVS Resource Definition (Online)* manual, and the DFHPCT macro is described in the *CICS/MVS Resource Definition (Macro)* manual. You should use this only for transactions which do not have their resources hard-coded within them and, for example, may accept the names of files as operator input. The CICS transactions CEBR, CEDF, CECI, CSMI, CSM1, CSM2, CSM3, and CSM5 are supplied with PCT definitions requiring resource checking. For more information on the CEBR, CEDF and CECI transactions, see the *CICS/MVS CICS-Supplied Transactions* manual. More information on the mirror transactions (CSMI, CSM1, CSM2, CSM3, and CSM5) is supplied in the *CICS/MVS Intercommunication Guide*.

The DFHEIP module makes a resource check against the resource-level key (SNNTASK) and the RSL value coded on the resource. This key is either acquired by signon or coded on the TCT definition. For information about coding RSL values for resources, you should read the *CICS/MVS Resource Definition (Online)* manual and *CICS/MVS Resource Definition (Macro)* manual.

All resources have a default value of 0. This implies no access, if RSLC=YES is specified. In particular, access to the resources of a remote system is denied.

Note: If your installation does not require stringent security checking, then you should not specify resource-level security checking for your transactions, and you should supply your own definitions of the CICS transactions CEBR, CEDF, CECI, CSMI, CSM1, CSM2, CSM3, and CSM5.

For information about security for EXEC CICS SET commands, you should read "Chapter 5.9. Examining and modifying resource attributes" on page 423.

Intercommunication link

When you establish an intercommunication LINK, a signon is performed. For an ISC connection, the name used for the signon is that specified by XSNAME=keyword on the DFHTCT TYPE=SYSTEM entry that is used to define the TCTTE for the intersystem LINK.

For an MRO or shared DB connection, the RACF name specified by USER=keyword in the job card is passed to the opposite system by the CICS SVC. If the SYSTEM entry does not specify XSNAME, then any connection will be accepted, and a RACF signon will be performed with the passed USER name. If the SYSTEM entry specifies an XSNAME, then only a connecting system with that USER name will be accepted.

In either case, a RACF signon is performed with the passed USER name or XSNAME value, and RACF will verify that this name is permitted to the CICS APPLID.

Intersystem security

Function shipping

The mirror transactions, CSMI and CVMI, execute in the remote system against this LINK TCTTE to honor the HLPI requests shipped from the local system. Resource-level checking is therefore made against the LINK TCTTE.

Note: If the mirror transaction is marked as requiring RACF checking, then you must PERMIT the USER name to access it and the resources on that system.

Transaction routing

The local PPT contains the program name DFHCRP. When the transaction is attached on the local system and passes the normal local security checks, this program will establish a link to the remote system. The remote transaction attach will be checked against the authority established in the LINK TCTTE at system-connect time. A pseudo TCTTE (surrogate TCTTE) will then be built and the transaction will run against this TCTTE. The surrogate TCTTE has no security. Therefore, any transaction that specifies RSLC(YES) will fail if used in transaction routing. However, see "DL/I program specification block," for PSB checking.

CRTE: terminal sharing

Using CRTE, the CICS-supplied routing transaction, on the local system establishes a link to the remote system and builds a surrogate TCTTE against which subsequent transactions will run. Security checks will be made against both the surrogate and LINK TCTTE. See "DL/I program specification block." You would normally sign on after issuing CRTE, so that the surrogate TCTTE would acquire some authority. Note that the signon runs in the remote system.

Asynchronous processing (START/RETRIEVE)

Use a START command to schedule a transaction on a remote system. If you specify a TERMID, then the TCTTE must have sufficient access authority to execute the transaction. The START command must be executed by the mirror transaction, and so the authority of the LINK TCTTE, as determined by the passed USER name or the XSNAME value, must be adequate.

For information about security in more than one system, see the *CICS/MVS Intercommunication Guide*.

DL/I program specification block

Since the PSB is not a CICS resource, RSL checking cannot be applied simply. A keyword PSBCHK on the SIT determines whether the RSLC keyword on the PCT for the transaction is to be honored for a surrogate TCTTE. PSBCHK=Y requests that the RSLC value be honored and PSBCHK=N that no check be performed. There is always a check against the LINK TCTTE. For nonterminal transactions and receive-only terminals, there is no security check.

Chapter 5.8. CICS monitoring facility

This chapter gives a brief overview of the CICS monitoring facility, then gives information on the following topics:

- Controlling the monitoring facilities
- Collecting the CICS monitoring data
- Defining event monitoring points for the collection of additional data
- Processing the output from the CICS monitoring facility
- User exits for accessing CICS monitoring data
- Collecting task-throughput data by means of RMF and SMF.

The CICS monitoring facility enables you to collect performance-related data during online processing for later offline analysis. The data is collected in three monitoring classes:

- **Accounting** class data. This is high-level information, such as the number of transactions for each combination of transaction identification and type, terminal identification, and user identification. This data can be used for installation accounting purposes.

- **Performance** class data. This is more detailed task-level information, such as the processor and elapsed time for a transaction, or the time spent waiting for I/O. This data can be used for performance monitoring and capacity planning. As well as data for each transaction, global performance data is collected, which provides information on the entire CICS system.

- **Exception** class data. This is information on exceptional conditions raised by a transaction, such as queuing for VSAM strings, or waiting for temporary storage. This data highlights possible problems in system operation.

Each class of monitoring can be active alone or jointly with other classes, and the information provided by the class data is described in the *CICS/MVS Performance Guide*.

Standard CICS monitoring data is collected at predefined event monitoring points (EMPs) in the CICS code. You can specify which classes of data are to be collected at these EMPs, but you have no control over the actual data collected. Timing information in the standard CICS monitoring data can be invalidated by transactions that terminate abnormally.

As well as the standard CICS monitoring data, user application programs can contribute data to user fields in the transaction-level accounting and performance class records but not the exception class records. In accounting class records, one field (a counter), is reserved for your use. In performance class records, one optional character-string field, and a variable number of fields for counters and time periods, can be used.

The monitoring data is collected in separate buffers for each monitoring class. The way in which these buffers are written to CICS user journals is controlled by the monitoring control table (MCT). It is possible to merge any two, or all three, classes of data onto one journal.

The monitoring data recorded in the CICS journal data sets can be analyzed later, either by using an IBM-supplied analysis program such as the Service Level Reporter (program number 5740-DC3), or by using a user-written analysis program. The CICS sample program DFH$MOLS, which lists the journal data set, can be used as a model for reading the data.

Controlling the monitoring facilities

The monitoring facilities are controlled by entries in the monitoring control table (MCT), the journal control table (JCT), the system initialization table (SIT), and by the CSTT transaction. Entries are also required in the CICS system definition (CSD) file if you are using resource definition online, or in the processing program table (PPT) and the program control table (PCT). You should read the information in the *CICS/MVS Operations Guide*, *CICS/MVS Resource Definition (Online)* manual, and *CICS/MVS Resource Definition (Macro)* manual for guidance on setting up and controlling the CICS monitoring facility.

Collecting CICS monitoring data

Most CICS monitoring facility data is initially collected on a transaction basis; a task monitoring area is reserved for this in the same dynamic storage block as the TCA, following the TWA, and preceding the TCA LIFO area. Storage for the monitoring area is allocated for monitoring classes that are active when the task starts.

CICS collects monitoring data only if monitoring and at least one of the appropriate monitoring classes are active *when the task starts*. You can activate monitoring and monitoring classes during a CICS session, but no monitoring data is collected for a task that has already started. For long-running tasks, like conversational or printer transactions, it is advisable to activate the required monitoring classes at startup.

You can deactivate monitoring and monitoring classes during a CICS session or you can leave CICS to deactivate monitoring. It does this during the first quiesce stage of system shutdown. No monitoring data is collected for tasks that start after that.

Monitoring data is then transferred to the buffers as appropriate to the class, usually at transaction end. For each class of CICS monitoring facility output, the MAXBUF operand of the MCT TYPE=RECORD macro specifies the size of CICS monitoring facility buffer to be reserved. This is allocated when that class of monitoring is activated either at startup (by an appropriately coded MONITOR operand in the SIT), or by the appropriate CSTT MONITOR transaction.

Monitoring data is accumulated in the buffers for each monitoring class, as described in the sections that follow.

Accounting class data

Accounting class data is accumulated for each separate combination of transaction identification, terminal identification, and eight-byte user identification (as established by CICS signon). As each transaction ends, its data is either added to an existing transaction-terminal-operator combination, or used to start another entry for a new combination.

One binary fullword, a counter field, is always reserved for user data. It is included in the accounting class records whether you modify it or not, and has a default value of zero.

Because the accounting records are so small, a MAXBUF of 2 or 4KB (kilobytes) is usually sufficient.

Performance class data

Performance class data is accumulated separately for each transaction and transferred to the buffer when the transaction ends. Two options of the DFHMCT TYPE=RECORD macro instruction control the collection of performance class data. The CONV=YES option specifies that the data for conversational transactions should be split up for each message pair. The CPU option specifies whether CPU usage and paging activity should be accumulated by transaction.

Each transaction performance record can be from 160 to 350 bytes long. The exact length depends on what user data is collected, and may be over 4000 bytes.

The following user fields will be reserved in the transaction-level performance class records if their existence is implied by DFHMCT TYPE=EMP macros coded in the MCT. For each distinct EMP qualifier:

- One or more fullword binary counts like the one in accounting class records. A maximum of 256 counts can be used.
- One or more clock fields. Each clock field consists of a fullword accumulator (an unsigned 32-bit counter) for periods of time in units of 16 microseconds, plus a separate binary fullword count of the number of separate time periods contributing to the total. A maximum of 256 clock fields can be used.

  Users familiar with the STCK doubleword TOD format will note that the unit of time in these clocks corresponds to the central four bytes of the STCK doubleword. The accumulator is a 32-bit **unsigned** counter, and thus has a maximum capacity of some 18 hours.
- One, and only one, byte-string field. You can, if you wish, regard this as a series of independent subfields. The maximum length of this field is 256 bytes.

Notes:

1. All the counts and clocks are initialized to binary zero. The byte-string field is initialized to nulls (X'00'), not blanks.
2. The value of the clock field for a clock that has been started and not stopped may be unexpected. CICS stops any running clocks before reporting them, and resets and restarts them immediately afterwards.

Exception class data

Exception class data is accumulated only for the transactions that encounter exception conditions; again, it is accumulated separately for each transaction, and is transferred to the buffer when the exception condition ends.

If a transaction encounters more than one type of exception, multiple exception records may be produced. If performance monitoring is also active, the corresponding transaction performance record will have a count of the number of exception records produced; all these records will have the task number in common.

Only one exception condition at one time can be reported for one task.

Notes:

1. Exception records for program compression are not output until one of the following occurs:

   - Task detach
   - A terminal conversation ends (CONV=YES)
   - Another exception record is output.

   This may result in more than one exception condition being reported in the same record. If multiple program compressions occur, only one record is produced, but the time-period count and exception-record count fields in the performance-class record are incremented for each occurrence of the exception.

2. An exception record for a suspend after a storage request may not be output until task-detach. If a task suffers this type of suspend more than once, the time-period (field SCWTETIM) may be accumulated because only one exception condition at a time can be reported for a task.

Global performance record

A single global performance record is maintained while the performance class is active. This record is written to the journal each time the performance class buffer for transactions is emptied due to the interval specified by the FREQ option on the DFHMCT TYPE=RECORD macro having expired, or due to the performance class being deactivated.

A user application program can add data to the accounting class records, and transaction-level performance class monitoring records, but not to the global performance class records or the exception class records.

Buffer requirements for monitoring

The monitoring data is sent to the CICS journal in separate sets of records, each set consisting of one of the following:

- One accounting class record.
- One performance class record.

- A group of exception class records, one for each exception condition recorded.
- One global performance class record.

Before each set is sent to the CICS journal buffer, it is prefixed with a section header and section descriptor. The length of the header-descriptor prefix is 38 bytes. In addition, the prefix has a string of field connectors. Because the number of the latter varies with the data class, the total length of the prefix ranges from 58 bytes for the accounting data to 140 bytes or more for transaction performance data.

In the journal buffer, space is reserved at the beginning of the block for the insertion of a block header just before the block is written out to the CICS journal. This block header contains a block sequence number, time written out, and so on. In an ordinary journal, the header would be a CICS journal label record. The total length is 58 bytes.

The format of the header and of the CICS data section are given later in this chapter.

All CICS data sections are journaled asynchronously, using DFHJC TYPE=WRITE requests without WAIT. They are then accumulated into blocks in the journal buffer, and are written out when that buffer is full, or when the buffer shift-up value (DFHJCT TYPE=ENTRY BUFSUV) is reached.

How large you need to make the CICS monitoring facility buffers depends on the emptying frequency chosen. If FREQ is low (but not zero), the buffers will be emptied frequently, and they can be kept small, although the journal buffers themselves should stay fairly large. This will even up the journal I/Os. Avoid making the CICS monitoring facility buffers too small, because you will use too much processing unit time when journaling the CICS monitoring facility data, and the proportion of CICS section header to actual data will become excessive.

When you decide on the sizes of journal buffers, do not forget that dictionaries, rather than monitoring data, might be the limiting factor. Values assigned to labels of the form MCTxx by the assembly of your MCT will show the buffer requirements for dictionary records.

Defining event monitoring points in user application programs

Occasionally, you may wish to gather additional information to that provided by the CICS monitoring facility. You can gather this information by specifying event monitoring points (EMPs) in your own programs. You can start and stop user-defined clocks, and update user counts at these points.

CICS provides additional user-defined counter and clock fields, to enable you to count the number of times a certain event occurs within an application, or within the total CICS system, and to enable you to time the interval between two events within an application, or within the CICS system.

To make use of these counters and clocks, the programmer defines EMPs where the CICS monitoring program is to collect additional information, in the application program or within program products that work in conjunction with your CICS system.

Trace (EXEC CICS ENTER) commands are used to define EMPs in CICS command-level application programs. DFHEMP macro instructions are used to define EMPs in CICS macro-level application programs.

Each EMP has an identification number. A user EMP in an application program will have no effect unless there is a corresponding entry for its identification number in the MCT and the appropriate monitoring classes are active.

The CICS monitoring facility does not provide user fields in either **exception** records or **global system performance** records. Application EMPs can therefore only record accounting and transaction-level performance data.

Command-level application programs

To code an EMP, you insert into your program the command-level request 'EXEC CICS ENTER TRACEID(...) MONITOR'. You specify an ID from 1 to 199. The MONITOR option indicates this is a monitoring EMP, not a normal user trace call. In addition, you have to specify either or both of the options ACCOUNT and PERFORM. ACCOUNT implies you will want to update user accounting data, PERFORM implies you will want to update performance data. Exactly what updates will be done depends on the matching TYPE=EMP entry in your MCT; if no such entry exists, the request will be ignored.

If you specify ACCOUNT or PERFORM without MONITOR, you cause a user trace entry as well as calling the CICS monitoring facility.

This EXEC CICS request also has an option 'FROM(...)', which enables you to pass data to the update process. 'FROM' should refer to eight bytes of contiguous storage, which will be regarded as two successive fullwords, DATA1 and DATA2. If you omit 'FROM(...)', fullwords consisting of binary zeros are passed instead.

IDs higher than 199 are reserved. In normal trace entries, IDs in the range 200-255 would be reserved for system trace calls; in monitoring they are reserved for use by other program products like DL/I. Thus the CICS-DL/I interface reserves 200-208.

The keyword 'ENTRYNAME(...)' may be used to qualify an ID. Thus TRACEID(3), for example, may be used both unqualified and qualified, by, for example, 'ENTRYNAME('UNIQUE')'. It becomes, effectively, two separate EMPs each managing its own set of clocks.

For details of the EXEC CICS ENTER command, see the *CICS/MVS Application Programmer's Reference* manual.

Macro-level application programs

User EMPs can be coded in assembler-language macro-level application programs using the macro DFHEMP TYPE=ENTRY with the CLASS operand. The syntax of the macro follows:

```
DFHEMP  [TYPE=ENTRY]
        ,ID=number|(PP,number)|entryname.number
        [,CLASS=([ACCOUNT][,PERFORM][,MONITOR])]
        [,DATA1={symbol|(symbol)]
        [,DATA2={symbol|(symbol)]
        [,RDATA1={register|(register)]
        [,RDATA2={register|(register)]
```

CLASS=([ACCOUNT][,PERFORM][,MONITOR])
Specifies that a user event monitoring point is to be defined.

ACCOUNT
Specifies that accounting class user data is to be collected at this user EMP if it is defined in the monitoring control table. ACCOUNT can be abbreviated to ACC.

PERFORM
Specifies that performance class user data is to be collected at this user EMP if it is defined in the monitoring control table. PERFORM can be abbreviated to PER.

MONITOR
Specifies that only a user EMP is to be defined at this point. MONITOR can be abbreviated to MON.

DATA1=
Specifies the address of the data to be placed in the first data field (bytes 8 to 11) of the trace table entry.

symbol
Is the symbolic address of the data to be placed in the first data field of the table entry.

(symbol)
Is the symbolic address of an area that contains the address of the data to be placed in the first data field.

DATA2=
Is similar to DATA1 except that it is used for the second data field (bytes 12 to 15) of the trace table entry.

ID=number|(PP,number)|entryname.number
Specifies the identification number of the event monitoring point. There may be an EMP-definition in the MCT that has the same ID=operand.

number
A decimal integer in the range 1 through 255.

(PP,number)
This form is used by some IBM program-products to generate IDs in the range 200 through 254 by specifying (PP,1) through (PP,55) respectively.

entryname.number
Allows multiple users of any number from 1 through 199. Thus, 'UNIQUE.3','DSN.3', and '3' would be three different EMP-identifiers. Any clocks, counts, or byte-offsets referred to by one of them will be different objects from those referred to by any other(s). If one of the forms 'number' or '(PP,number)' is coded without an entryname, a default entryname 'USER' is provided.

RDATA1=
Specifies the register whose contents are to be placed in the first data field of the trace table entry.

register
The number of the register whose contents are to be placed in the first data field.

(register)
The number of the register whose contents are the address of the data to be placed in the first data field.

RDATA2=
Is similar to RDATA1, except that it is used for the second data field of the trace table entry.

For monitoring, the fields DATA1 and DATA2 identify optional binary fullwords that can represent a number, a mask for logical operations, or an address. The way in which DATA1 and DATA2 are interpreted will depend on the update actions requested in the monitoring control table.

If both the ACCOUNT and PERFORM options are specified in the application program, the corresponding MCT entry can specify recording of either accounting or performance data, or both. If only one option is specified at the user EMP, only that class of recording is possible. Thus, greater flexibility is obtained by specifying both options for the user EMP, and controlling run-time activity by a suitably coded MCT.

Specifying event monitoring point processing in the MCT

Throughout this section, refer to the chapter on the DFHMCT macro instruction in the *CICS/MVS Resource Definition (Macro)* manual.

DFHMCT TYPE=EMP

The DFHMCT TYPE=EMP macro instruction describes what action is to be taken at each user-defined event monitoring point (EMP). The actions that can be specified include manipulating the user count field in the accounting class and performance (transaction) class records, and the user clock field and user byte field in the performance (transaction) class records. One DFHMCT TYPE=EMP macro instruction must be coded for each EMP to be used.

Class (CLASS) fields

CLASS=ACCOUNT|PERFORM specifies which of the monitor classes are to be manipulated for this EMP. ACCOUNT specifies that accounting class data is to be changed; PERFORM specifies that performance class data is to be changed. One or both values must be specified.

If CLASS=ACCOUNT is coded, the ACCOUNT= operand (described under "ACCOUNT and PERFORM fields") is used to define the specific actions to be taken to adjust the accounting class user count field.

If CLASS=PERFORM is coded, the PERFORM= operand (described under "ACCOUNT and PERFORM fields") is used to define the specific actions to be taken to adjust the performance class user count, user clock, and user byte fields.

ID (EMP identifier) field

ID=number|(PP,number)|entryname.number specifies the identification number of the EMP for which this TYPE=EMP macro instruction applies.

For EMPs within a user application program, you should specify the ID as a number from 1 through 199. Identification numbers between 200 and 255 are reserved for EMPs within program products, and these are coded as ID=(PP,n), which is equivalent to ID=199+n. The CICS-DL/I interface, for example, uses ID=(PP,1) up to ID=(PP,9). If relevant EMPs have been defined in the code of a program product, the EMP identifiers will be given in the documentation for that product.

The "entryname.number" form is described under the DFHEMP macro in the preceding section.

ACCOUNT and PERFORM fields

If you have coded CLASS=ACCOUNT, code ACCOUNT=(option) to define the actions to be taken against the user count field in the accounting class data record for this EMP. See the *CICS/MVS Resource Definition (Macro)* manual for the options available for this operand.

If you have coded CLASS=PERFORM, code PERFORM=(option) to define the actions to be taken against the user count, user clock, and user byte fields in the performance class data record for this EMP. See the *CICS/MVS Resource Definition (Macro)* manual for the options available for this operand.

The user fields consist of:

1. Up to 256 counters;
2. Up to 256 clocks, each made up of a four-byte accumulator and four-byte count; and
3. A byte string of up to 256 bytes.

When you are selecting the user clock and user count fields for use with your EMPs, use the lowest-numbered fields that are available. When it is activated, the CICS monitoring facility acquires from dynamic storage a buffer large enough to hold the largest possible monitor record. The internal calculations for the size of this buffer are based on the field number. More storage will (perhaps unnecessarily) be allocated for a higher-numbered field than for a lower-numbered field, on the assumption that all lower-numbered fields are already in use.

Clock (SCLOCK and PCLOCK) fields (performance data only)

SCLOCK(number)
The clock specified by number is to be started. The value of the four-byte count in the user clock field will be incremented by 1, and flagged to indicate "running".

PCLOCK(number)
The clock specified by number is to be stopped. The value of the four-byte count in the user clock field will have its flag reset. The accumulator is reset to the sum of its contents before the previous SCLOCK and the elapsed period between that SCLOCK and this PCLOCK.

An EMP causes a clock to be started if the corresponding MCT entry specifies SCLOCK. At this point, the actual clock contents in the transaction performance record are set to the start TOD in units of 16 microseconds, backdated by the total time accumulated so far, and the count of periods accumulated so far is flagged, to indicate that the clock is running.

When, at another EMP, the corresponding PCLOCK option is executed, the clock contents are subtracted from the stop TOD to give the total accumulated time. At the same time, the halfword count of contributing periods has its flag reset (to indicate that the clock is stopped).

In some circumstances, a transaction may end with an outstanding SCLOCK request (for example, because of abend). In such cases, the clock contents will be misleading, but the fact will be highlighted by the period count associated with the clock being flagged (high-order bit set to 1). If this is undesirable, users can code a suitable DFHPEP abend exit to test for this in vital clocks and, if necessary, to close off the last period, as if PCLOCK had been issued at task end.

Starting and stopping clocks does not refer to the data passed as DATA1 and DATA2 from the EMP, so clocks can be controlled at the same time as counts are updated.

At some cost in collecting the data, you can use clocks to accumulate the time that the CICS main task TCB has been running (CPU-time) rather than total elapsed time. You do this by using the options SCPUCLK and PCPUCLK instead of SCLOCK and PCLOCK respectively.

Sample DFHMCT TYPE=EMP entries for DL/I

The following sample MCT entries make use of the nine event monitoring points in the PERFORM class used by the CICS-DL/I interface modules:

```
DFHMCT TYPE=EMP,ID=(PP,1),CLASS=PERFORM,PERFORM=SCLOCK(3)
DFHMCT TYPE=EMP,ID=(PP,2),CLASS=PERFORM,PERFORM=PCLOCK(3)
DFHMCT TYPE=EMP,ID=(PP,3),CLASS=PERFORM,PERFORM=SCLOCK(1)
DFHMCT TYPE=EMP,ID=(PP,4),CLASS=PERFORM,PERFORM=(PCLOCK(1)
                                        ,SCLOCK(2))
DFHMCT TYPE=EMP,ID=(PP,5),CLASS=PERFORM,PERFORM=(PCLOCK(2)
                                        ,MLTCNT(1,9))
DFHMCT TYPE=EMP,ID=(PP,6),CLASS=PERFORM,PERFORM=SCLOCK(4)
DFHMCT TYPE=EMP,ID=(PP,7),CLASS=PERFORM,PERFORM=PCLOCK(4)
DFHMCT TYPE=EMP,ID=(PP,8),CLASS=PERFORM,PERFORM=(PCLOCK(1)
                                        ,PCLOCK(2),PCLOCK(3)
                                        ,PCLOCK(4))
DFHMCT TYPE=EMP,ID=(PP,9),CLASS=PERFORM,PERFORM=PCLOCK(1)
```

In the ID parameter, the following specifications are used:

```
(PP,1) - Start of database call
(PP,2) - End of database call
(PP,3) - Start of PSB schedule
(PP,4) - End of schedule call
(PP,5) - Terminate PSB
(PP,6) - Start of I/O wait
(PP,7) - End of I/O wait
(PP,8) - Stop all clocks after task abend in DFHDLQ.
(PP,9) - Stop clock 1 after schedule failure
```

Processing output from the CICS monitoring facility

Output from the CICS monitoring facility can be processed in the following ways:

You can code your own output-processing and analysis program. The information in the sections, "CICS monitoring data formats" on page 399, "Data records produced by CICS monitoring" on page 406, and "User exits for accessing monitoring data" on page 416 will help you to do this.

You can use the sample program DFH$MOLS, described below. This program generates a basic listing of the CICS monitoring facility output. You may wish to use DFH$MOLS to check that the monitoring facility has been correctly specified, or you may wish to modify it to provide your own analysis programs. If you reassemble DFH$MOLS using a COPY statement, you will need to include the END DFH$MONA statement in your assembly JCL.

Release 2 of the OS/VS program product Service Level Reporter (program number 5740-DC3) runs under MVS, and can be used to process CICS monitoring output, combining it with other performance information from, for example, system management facilities (SMF), and producing joint reports. CICS users with SLR may, therefore, wish to use the DFHJCT JTYPE=SMF option, combining CICS monitoring facility output with other SMF information on standard SMF data sets, rather than sending CICS monitoring facility output to a separate CICS journal.

The DFH$MOLS program

The sample program DFH$MOLS lists the journal data sets produced by the CICS monitoring facilities. It does not analyze the data. Information on how to run the sample program is given in the source of the program, which should be listed from the appropriate system source library.

The output from the sample program is a list of fields and field values from the journal data set. For each block of data on the journal, field values are listed from the SMF header and product section. For each CICS data section, section header and descriptor field values are listed, followed by the field description and value for each data field present in the first data record. The field descriptions and values are then repeated for the remaining data records in the data section.

Function

The DFH$MOLS program reads, formats, and prints the monitoring data set, that is, the system SMF data set or the CICS journal. It is intended as a sample program that you can use as a skeleton if you need to write your own program to analyze the data set.

This program can also be read with the monitoring documentation to enable you to understand the "wrapping" that encloses the raw monitoring data. After the "wrapping" has been removed, the individual data fields are moved to the print area. If you wish to analyze the data, this would be the point that the link to the user exit would be made.

Monitoring manipulates counters, clocks, and fields. Clocks may be four-byte elapsed time or eight-byte time-of-day (TOD). The program converts the TOD clock to yy/mm/dd format.

The program needs to find a dictionary. DFHCMON constructs the dictionary automatically, in each journal, when monitoring is started. If the data set does not contain a dictionary, the program does not analyze any records but simply abends.

This program produces about a page of output for each task if all classes of monitoring are active.

The monitoring data set on disks should be closed before the DFH$MOLS program is run. This ensures that an EOF marker is written and avoids processing information written previously.

The monitoring data set is expected to be in the following format:

```
<SMF header>.<product section>.<CICS section>.<CICS section>
```

There can be many occurrences of the CICS section. The format of the <CICS section> is:

```
<function header>.<section descriptor>(<section data>|<section dict>)
```

This program requires the section dictionary to occur before any section data.

It is the user's responsibility to close the journal data sets.

Operation

The program reads the monitor data set until the dictionary records are located. It then builds in-core dictionaries, and processes the following data records against them. If more dictionary records are found, the current in-core ones are released and new ones built.

Selectivity and sample JCL

An optional selection argument, specified in the PARM-field of the execution JCL, is interpreted by the program as a concatenation of transaction-and terminal-identifiers. When correctly specified, the program will suppress the printing of dictionary information, of record headers, and of all data not associated with the specified combination of transaction and terminal.

The example JCL is as follows:

```
//J1  JOB
//S1  EXEC PGM=DFH$MOLS[,PARM=...]
//INPUT  DD (input tape or disk)
//OUTPUT DD SYSOUT=A
```

CICS monitoring data formats

This section gives details of the formats of CICS monitoring-facility records. However, you should be aware that these formats may change due to maintenance or development activity of CICS. Look in the *OS/VS2 MVS Supervisor Services and Macro Instructions*, GC28-0683, for the format of the records written to SMF data sets.

When output data from the CICS monitoring facility is processed, the first block for any class that an analysis program encounters is a dictionary record. Dictionary formats are described under "Dictionary section" on page 405.

The output of the CICS monitoring facilities is in standard System Management Facilities (SMF) format (record type '110').

Each physical block is composed of a standard block header (SMF header and product section), and then one or more CICS data sections; each CICS section comprises a section header and descriptor, and then one or more CICS monitoring-facility data records. Within each section, all the CICS monitoring-facility data records have the same format. Figure 34 on page 400 shows the format of a block of monitoring data.

The dictionary information for each monitoring class is contained in CICS dictionary sections.

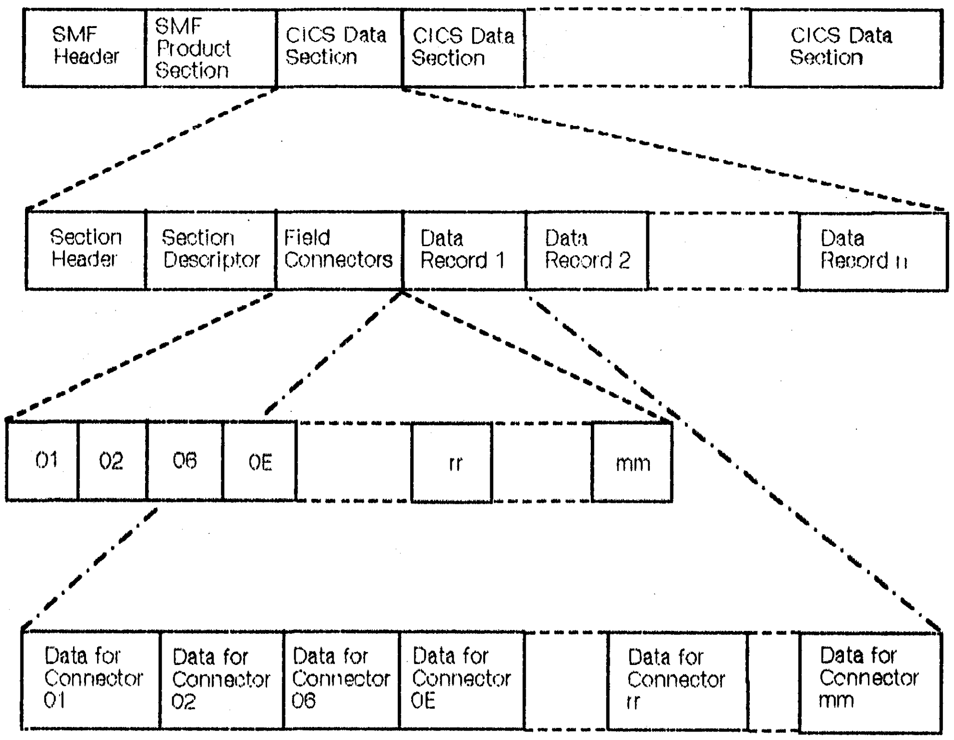


Figure 34. Format of a block of monitoring data

Standard system management facilities block header

The standard system management facilities (SMF) block header consists of three parts: a four-byte "LLbb" block length field, an SMF header, and a product section. The SMF header describes the system creating the output; the product section describes where the output originated, that is, the subsystem producing the output (CICS region), the log ID, and so on. Many of the fields are unimportant for CICS monitoring-facility output to user journals; but in MVS, where the CICS monitoring-facility data may be directed to SMF data sets and merged with other SMF data, it must be possible to distinguish the types of data during processing by standard SMF-based programs.

The total length of the containing physical block is specified in the initial LLbb field. The remaining fields are as described below. The names listed on the left can be generated within a user-defined DSECT, by coding the macro DFHJCSMF TYPE=(PREFIX,SMF). For an example, see "Example of SMF block mapping" on page 404. The offsets shown must be increased by 4 when reckoned from the LLbb field at the start of the block.

SMF header

The SMF header describes the system creating the output and has the following format:

Field DSECT Name	Offset (Hex)	Length (Bytes)	Description
SMFFLG	0	1	Operating system indicator X'82' (SMFFLVS2)
SMFRTY	1	1	SMF record type — always X'6E' for CICS
SMFTME	2	4	Time when block written (binary time-of-day clock in 0.01-second units)
SMFDTE	6	4	Day-of-year date when block written — packed decimal 00YYDDD+
SMFSID	A	4	System identification
SMFSSS	E	4	Subsystem identification
	12	2	Reserved
SMFAPS	14	4	Offset of start of product section (SMFPSSTY) from block LLbb prefix
SMFLPS	18	2	Length of product section
SMFNPS	1A	2	Number of product sections

SMF product section

The product section identifies the subsystem producing the output, which, in this case, is the CICS region. It has the following format (offsets are from the start of the SMF header):

Field DSECT Name	Offset (Hex)	Length (Bytes)	Description
SMFPSSTY	1C	2	Product subtype — always H'1' for monitoring
SMFPSRVR	1E	2	Record version — C'03'
SMFPSPRN	20	8	Recording product name — APPLID known to VTAM
SMFPSRSN	28	4	Block sequence number in this journal — seven-digit packed decimal
SMFPSJID	2C	1	Journal identification — user journal ID for monitoring
SMFPSBKN	2D	3	Record number within data set
SMFPSLBW	30	4	Last record address (relative track and record TTR)
SMFPSBAL	34	2	Track balance in bytes

CICS data section

Each CICS data section consists of a section header, a section descriptor, a string of field connectors, and one or more data records of identical format relating to one monitoring class.

The four different types of CICS data section are distinguished by the values of two fields: the class field (MNSEGCL) in the section header and the record type (MCTSSDID) in the section descriptor.

After the section descriptor, and before the section data records, is a string of binary field connectors that indicate, in order, the fields present in each data record. Each connector corresponds to a specific field-entry in the associated dictionary (described under "Dictionary section" on page 405), which supplies the field length, format, and name. Thus, the string of field connectors, coupled with the dictionary, can be used by reporting programs to analyze each data section. In fact, this is the **only** way a reporting program can interpret the data sections with some independence from changes introduced by maintenance or new releases.

In records of transaction data, the successive records can be regarded as rows in a table, each column corresponding to one type of field within the records. Each field connector then describes the contents of one column. This view of the data is helpful when designing tabular reports, which are often arranged in just this way.

Section header

Each section header begins with a four-byte "LLbb" length field, which defines the overall length of the entire CICS section, including this prefix, the section header, the section descriptor, the field connectors, and the data. This is followed by a halfword containing the length of the section header (including this halfword but excluding the preceding LLbb).

The remaining fields in the section header are as described below. The names listed on the left can be generated within a user-defined DSECT by coding the DFHCMPRC TYPE=(PREFIX,MNS) macro. For an example, see "Example of SMF block mapping" on page 404. The offsets shown should be increased by 6 when reckoned from the LLbb at the start of the section header.

Field DSECT Name	**Offset (Hex)**	**Length (Bytes)**	**Description**
MNSEGCL	0	2	Class of data captured H'2' – accounting data H'3' – performance data H'4' – exception data
MNSEGSYS	2	8	Subsystem ID – APPLID known to VTAM
MNSEGID	A	4	Machine on which data was produced – always CICS
MNSEGREG	E	2	Reserved

Section descriptor

The format of the section descriptor follows. The field names can be generated in a user-defined DSECT using the macro DFHMCTSS TYPE=(PREFIX,MCT). For an example, see "Example of SMF block mapping" on page 404.

Field DSECT Name	Offset (Hex)	Length (Bytes)	Description
MCTSSDL	0	2	Total length of section descriptor and section data
MCTSSDID	2	2	Type of record within class H'0' – dictionary data H'1' – transaction data H'2' – global data (performance only)
MCTSSDCA	4	2	Offset from MCTSSDL to start of field connectors
MCTSSDCL	6	2	Length of each field connector
MCTSSDCN	8	2	Number of connectors
MCTSSDRA	A	2	Offset from MCTSSDL to start of first data record
MCTSSDRL	C	2	Length of each data record
MCTSSDRN	E	2	Number of data records

Field connectors and data records

The section descriptor is followed by the string of field connectors, and then by the successive data records. The connector value is equal to the value of the CMODCONN field (that is, the assigned connector value) in the corresponding dictionary entry. Each field connector in the string preceding the data records corresponds with one field in each record; the first field connector with the first field, the second with the second, and so on.

There is one dictionary section for all records in a monitoring class. In the case of the performance class, one set of field connectors serves for both transaction and global performance records. The string of connectors associated with each type of record is a subset of the complete range.

In the case of the user clocks, one connector is used to designate the entire eight bytes that contain the time accumulator and the period count.

Example of SMF block mapping

The following code shows how an SMF block can be mapped using the macros mentioned in the descriptions of the SMF block header and the CICS data section. This example takes care of the offset-adjustments referred to in those descriptions. All labels beginning 'XXX' may be replaced by your own labels. Your replacement label must have three characters. If it does not, the assembly will be in error. If the prefix is omitted, the default is given.

```
JCSMFDS  DSECT ,                    Must use the name JCSMFDS
XXXBLK   DS    HL2                  Length of block (LL)
         DS    HL2                  Filler (BB)
         DFHJCSMF TYPE=(PREFIX,SMF)
*                                   Defines remainder of SMF-header
*                                   and product-section
XXXSEG   DSECT ,
XXXSEC   DS    HL2                  Length of CICS data-section (LL)
         DS    XL2                  Filler (BB)
         DS    HL2                  Length of section header
*        (Note that halfword definitions in the following
*        structures may become unaligned at execution-time).
         DFHCMPRC TYPE=(PREFIX,MNS)
*                                   Defines CICS section-header
         DFHMCTSS TYPE=(PREFIX,MCT)
*                                   Defines CICS section-detail
*          ...
XXXREG   EQU   3                    (Choose your own register)
XXXCSECT CSECT ,
*          ...
*              Assuming that XXXREG already addresses the block, ...
         USING JCSMFDS,XXXREG       Tell Assembler
*          ...
*        Retrieve what you need from the SMF-header (especially the
*                          blocklength from XXXBLK), and from the
*                          product-section then ...
*          ...
         LA    XXXREG,SMFEND        Address 1st CICS data-section
         USING XXXSEG,XXXREG        Tell Assembler
*              (Addressability to SMF-header is now lost, since
*              this example uses just the one register - XXXREG).
*          ...
*        Now MNSEGCL through MNSEGREG show the origin of the data,
*        and MCTSSDL through MCTSSDRN define its format. Processing
*        here probably consists of iterations, controlled by the
*        value of MCTSSDRN, interpreting each data-record accord-
*        ing to the field-IDs defined by MCTSSDCA/L/N.
*            ...
         AH    XXXREG,XXXSEC        Address next CICS data-section
*              (This is where to test for end-of-block by reference
*                  to the value derived from the blocklength field).
```

Dictionary section

Each of the three classes of monitoring data has a dictionary section to describe the data format and length of each field within the class records, and to provide each field with a descriptive title.

Each dictionary section consists of a section header and section descriptor followed by dictionary data records.

You can use the DFHCMPDR DIR,,[ACC|PER|EXC] macro to list the dictionary contents for each monitoring class.

Dictionary section descriptor

The format of a dictionary section descriptor follows.

Field DSECT Name	Offset (Hex)	Length (Bytes)	Description
MCTSSDL	0	2	Total length of section descriptor and dictionary data
MCTSSDID	2	2	Type of record within class H'0' – dictionary data
MCTSSDCA	4	2	Undefined for dictionary data
MCTSSDCL	6	2	Undefined for dictionary data
MCTSSDCN	8	2	H'0' – dictionary data
MCTSSDRA	A	2	Offset from MCTSSDL to start of dictionary data
MCTSSDRL	C	2	Length of each dictionary entry
MCTSSDRN	E	2	Number of dictionary entries

Dictionary sections do not have the string of field connectors that are included in data sections. Instead, they describe the format and size of the field associated with each connector, by repeating an "entry", the contents of which can be mapped by the macro DFHCMPDR TYPE=(PREFIX,CMO). If you wish to replace CMO with your own label, you must supply a three-character label. If you do not, you will receive the default.

Offset (Hex)	Length (Bytes)	Name	Description
0	CL8	CMODNAME	Owner of the field
8	CL1	CMODTYPE	Data type: 'S' Clock 'T' Time-stamp 'A' Count 'C' Byte-string 'P' Packed decimal
9	CL3	CMODIDNT	Numeric name
C	HL2	CMODLENG	Length of data
E	HL2	CMODCONN	Assigned connector
10	CL8	CMODHEAD	Informal field name

Figure 35 shows the format of a block of dictionary data.

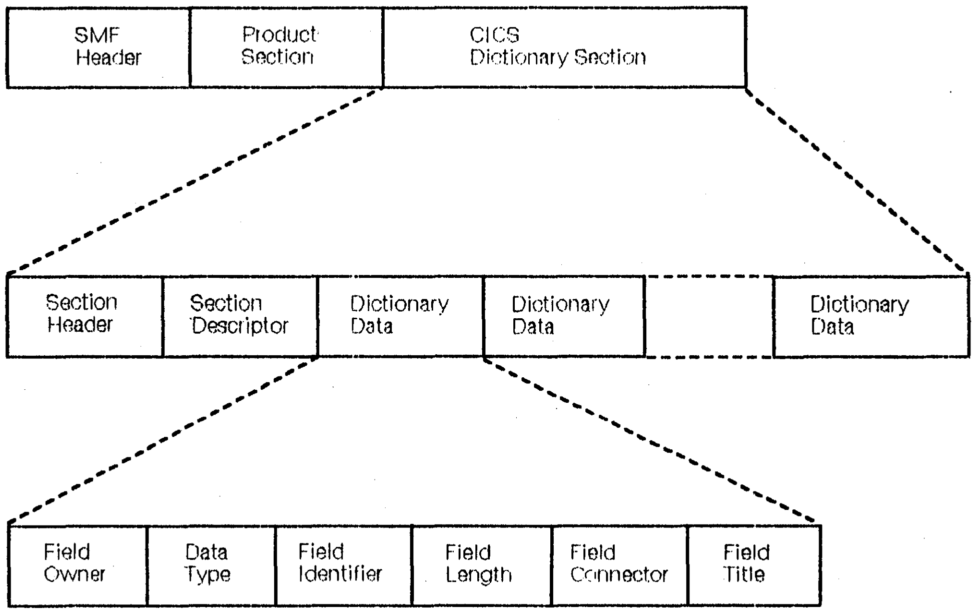


Figure 35. Format of a block of dictionary data for monitoring

The field DSECT names in the section header and section descriptor are the same as for a CICS data section and can be generated using the DFHMCTSS and DFHCMPRC macro instructions. However, when no string of field connectors is present, some fields in the section descriptor have fixed values. For example, MCTSSDCN=X'0000'.

Data records produced by CICS monitoring

All the data fields that are available in monitoring data records for all the monitoring classes are shown below. Each field is associated with a field connector that links it to its dictionary entry. The dictionary entry for each field defines the length and format of the data for that field, and contains a descriptive title.

The definitive descriptor associated with any field connector is that contained in the dictionary. The dictionary produced when the monitoring data is journaled defines the data it accompanies.

This description divides the monitoring data into three classes of record as indicated by MNSEGCL in the CICS section header. For each class, the various fields are described briefly in order of their field connectors. Note that this order is not the same as their physical order when journaled.

In the descriptions that follow, the term **clock** is distinguished from **time stamp**. A **clock** comprises a 32-bit value, expressed in units of 16 microseconds, accumulated during one or more measurement periods, followed by eight reserved bits, in turn followed by a 24-bit value indicating the number of such periods. A **time stamp**, on the other hand, is an eight-byte copy of the output of a store clock (STCK) instruction.

Neither the 32-bit timer component of a **clock**, nor its 24-bit period count are protected against wraparound. The timer capacity is about 18 hours, and the period count runs modulo-16777216. The "reserved bits" have the following significance:

Bits 0 and 1
are used for online control of the clock when it is running, and should always be zeros on output.

Bits 2 and 3, 4 and 7
are not used.

Bits 5 and 6
when set to 1, indicate that the clock has suffered at least one out-of-phase start (bit 5) and/or stop (bit 6).

Note that the phrase **waiting for ... I/O**, which occurs in the record descriptions, includes not only that time during which an I/O operation is actually taking place, but also the time in which the access method is completing the outstanding event control block and subsequently until the waiting CICS task is redispatched.

Monitoring-field definitions

These definitions are each presented as a basic field name followed by a definition of the contents. The name has the format shown in this example:

```
001 DFHTASK (TYPE-C, 'TRAN', 4 BYTES.)
 |     |        |       |       |
 |     |        |       |      Length of the field (as represented by
 |     |        |       |      CMODLENG in the relevant dictionary entry).
 |     |        |       |
 |     |        |      Informal name for the field, as used, perhaps, in
 |     |        |      column headings when the monitoring output is
 |     |        |      post-processed. (CMODHEAD of the dictionary entry).
 |     |        |
 |     |       (This is CMODTYPE of the dictionary entry).
 |     |       The following types are defined:
 |     |        A - a 32-bit count
 |     |        C - a byte-string
 |     |        P - a packed decimal value
 |     |        S - a clock comprising a 32-bit accumulation of 16-microsecond
 |     |            units followed by a 32-bit count (modulo-16,777,216) of
 |     |            the number of intervals included in the accumulation.
 |     |        T - a time stamp derived directly from the output of a
 |     |            store clock (STCK) instruction.
 |     |
 |    Group name by which multiple fields may be EXCLUDEd or INCLUDEd
 |    during MCT preparation.
 |    (Field CMODNAME of the relevant dictionary entry).
 |
Field identifier (unique within class) by which the field may
be individually EXCLUDEd or INCLUDEd during MCT preparation.
(Field CMODIDNT of the relevant dictionary entry).
```

Accounting class

001 DFHTASK (TYPE-C, 'TRAN', 4 BYTES.)
Transaction identification.

002 DFHTERM (TYPE-C, 'TERM', 4 BYTES.)
Terminal identification.

004 DFHTASK (TYPE-C, 'T', 4 BYTES.)
Transaction type:

A	Attached by automatic transaction initiation (ATI)
D	Attached by transient-data trigger-level
S	System internal task
T	Attached from a terminal (but see 'Z' below)
Z	Second or subsequent part of a pseudoconversation.

005 DFHTASK (TYPE-A, 'OCCURS', 4 BYTES.)
Number of tasks identified by fields 1, 2, 11, 4.

006 DFHTASK (TYPE-A, 'ABENDS', 4 BYTES.)
Number of tasks that abended.

007 DFHTERM (TYPE-A, 'MSGIN', 4 BYTES.)
Total number of input messages.

008 DFHCICS (TYPE-A, 'UCOUNT', 4 BYTES.)
User count field.

009 DFHCICS (TYPE-T, 'START', 8 BYTES.)
Earliest task-start time.

010 DFHCICS (TYPE-T, 'STOP', 8 BYTES.)
Most recent task-detach time.

011 DFHCICS (TYPE-C, 'USERID', 8 BYTES.)
User identification.

Performance class

In the performance class, a user-task (lifetime of a TCA) may be represented by one or more monitoring records, according to whether the MCT option DELIVER and/or the MCT parameter CONV=YES have been selected. In the descriptions that follow, the term "user-task" can usually be interpreted to mean "that part or whole (of a TCA-lifetime) that is represented by a performance class record", unless the description specifies otherwise.

The phrase "during the measurement interval" normally identifies a field as being part of the global performance record. Note that a few fields are common to global and task-level records.

001 DFHTASK (TYPE-C, 'TRAN', 4 BYTES.)
Transaction identification.

002 DFHTERM (TYPE-C, 'TERM', 4 BYTES.)
Terminal identification.

003 DFHCICS (TYPE-C, 'OPR', 4 BYTES.)
User identification.

004 DFHTASK (TYPE-C, 'T', 4 BYTES.)
Transaction type:

A	Attached by automatic transaction initiation (ATI)
C	Second or subsequent part of a conversational task
D	Attached by transient-data trigger-level
S	System internal task
T	Attached from a terminal (but see 'Z' below)
Z	Second or subsequent part of a pseudoconversation.

005 DFHCICS (TYPE-T, 'START', 8 BYTES.)
Start-time of measurement interval. For global records, this is time at which the interval began, as determined by the FREQ= parameter in the performance class RECORD entry of the MCT. For task-level records, this is either the time at which the user-task was attached, or the time at which data-recording was most recently reset in support of CONV=YES or of DELIVER.

006 DFHCICS (TYPE-T, 'STOP', 8 BYTES.)
Finish-time of measurement interval. For global records, this is time at which the interval ended, as determined by the FREQ= parameter in the performance class RECORD entry of the MCT. For task-level records, this is either the time at which the user-task was detached, or the time at which data-recording was completed in support of CONV=YES or of DELIVER.

007 DFHTASK (TYPE-S, 'USRDISPT', 8 BYTES.)
Elapsed time for which the user-task was dispatched.

008 DFHTASK (TYPE-S, 'USRCPUT', 8 BYTES.)
Processor-time for which the user-task was dispatched.

009 DFHTERM (TYPE-S, 'TCIOWTT', 8 BYTES.)
Elapsed time for which the user-task waited for terminal I/O.

010 DFHJOUR (TYPE-S, 'JCIOWTT', 8 BYTES.)
Elapsed time for which the user-task waited for journal I/O.

011 DFHTEMP (TYPE-S, 'TSIOWTT', 8 BYTES.)
Elapsed time for which the user-task waited for temporary-storage I/O.

014 DFHTASK (TYPE-S, 'SUSPTIME', 8 BYTES.)
Elapsed time for which the user-task was on the suspend-chain.

015 DFHTERM (TYPE-S, 'TCDISPT', 8 BYTES.)
Elapsed time for which the terminal manager was dispatched during the measurement interval.

016 DFHTERM (TYPE-S, 'TCCPUT', 8 BYTES.)
Processor-time for which the terminal manager was dispatched during the measurement interval.

018 DFHJOUR (TYPE-S, 'JCDISPT', 8 BYTES.)
Elapsed time for which the journal manager was dispatched during the measurement interval.

019 DFHJOUR (TYPE-S, 'JCCPUT', 8 BYTES.)
Processor-time for which the journal manager was dispatched during the measurement interval.

021 DFHTASK (TYPE-S, 'USRDISPT', 8 BYTES.)
Elapsed time for which any user-task was dispatched during the measurement interval.

022 DFHTASK (TYPE-S, 'USRCPUT', 8 BYTES.)
Processor-time for which any user-task was dispatched during the measurement interval.

024 DFHTASK (TYPE-S, 'KCDISPT', 8 BYTES.)
Elapsed time for which CICS's task-manager was dispatched during the measurement interval.

026 DFHSTOR (TYPE-A, 'DSAHWM', 4 BYTES.)
Maximum amount (high-water mark) in pages of the dynamic storage area used during the measurement interval.

027 DFHTASK (TYPE-S, 'OSWTELAT', 8 BYTES.)
Elapsed time spent by the CICS main task in operating system WAIT(s) during the measurement interval.

028 DFHTASK (TYPE-S, 'SRBTIME', 8 BYTES.)
Time spent by CICS in SRB-mode during the measurement interval. Instead of the actual number of times SRB-mode was entered, the entry-count

portion of this clock always contains a value of 1. This is because, in order to reduce overhead, the operating system SRB-timer for the address-space is interrogated only at the end of the interval.

029 DFHSTOR (TYPE-A, 'PAGINCT', 4 BYTES.)
Page-in count during the measurement interval.

030 DFHSTOR (TYPE-A, 'PAGOUCT', 4 BYTES.)
Page-out count during the measurement interval.

031 DFHTASK (TYPE-P, 'TASKNO', 4 BYTES.)
Sequence number of the TCA (TCAKCTTA).

032 DFHCICS (TYPE-A, 'MNEXCCT', 4 BYTES.)
Number of exception records generated by the user-task.

033 DFHSTOR (TYPE-A, 'SCUSRHWM', 4 BYTES.)
Maximum amount (high-water mark) of storage allocated to the user-task. In an MVS/XA system, this value includes storage obtained above 16MB.

034 DFHTERM (TYPE-A, 'TCMSGIN1', 4 BYTES.)
Number of messages received from the principal terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

035 DFHTERM (TYPE-A, 'TCMSGOU1', 4 BYTES.)
Number of messages sent to the principal terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

036 DFHFILE (TYPE-A, 'FCGETCT', 4 BYTES.)
Number of file GETs issued by the user-task.

037 DFHFILE (TYPE-A, 'FCPUTCT', 4 BYTES.)
Number of file PUTs issued by the user-task.

038 DFHFILE (TYPE-A, 'FCBRWCT', 4 BYTES.)
Number of file BROWSEs issued by the user-task.

039 DFHFILE (TYPE-A, 'FCADDCT', 4 BYTES.)
Number of file ADDs issued by the user-task.

040 DFHFILE (TYPE-A, 'FCDELCT', 4 BYTES.)
Number of file DELETEs issued by the user-task.

041 DFHDEST (TYPE-A, 'TDGETCT', 4 BYTES.)
Number of transient-data GETs issued by the user-task.

042 DFHDEST (TYPE-A, 'TDPUTCT', 4 BYTES.)
Number of transient-data PUTs issued by the user-task.

043 DFHDEST (TYPE-A, 'TDPURCT', 4 BYTES.)
Number of transient-data PURGEs issued by the user-task.

044 DFHTEMP (TYPE-A, 'TSGETCT', 4 BYTES.)
Number of temporary-storage GETs issued by the user-task.

046 DFHTEMP (TYPE-A, 'TSPUTACT', 4 BYTES.)
Number of PUTs to auxiliary temporary-storage issued by the user-task.

047 DFHTEMP (TYPE-A, 'TSPUTMCT', 4 BYTES.)
Number of PUTs to main temporary-storage issued by the user-task.

050 DFHMAPP (TYPE-A, 'BMSMAPCT', 4 BYTES.)
Number of BMS MAP requests issued by the user-task.

051 DFHMAPP (TYPE-A, 'BMSINCT', 4 BYTES.)
Number of BMS IN requests issued by the user-task.

052 DFHMAPP (TYPE-A, 'BMSOUTCT', 4 BYTES.)
Number of BMS OUT requests issued by the user-task.

054 DFHSTOR (TYPE-A, 'SCUGETCT', 4 BYTES.)
Number of user-storage GETMAINs during the user-task.

055 DFHPROG (TYPE-A, 'PCLINKCT', 4 BYTES.)
Number of program LINKs during the user-task.

056 DFHPROG (TYPE-A, 'PCXCTLCT', 4 BYTES.)
Number of program XCTLs during the user-task.

057 DFHPROG (TYPE-A, 'PCLOADCT', 4 BYTES.)
Number of program LOADs during the user-task.

058 DFHJOUR (TYPE-A, 'JCPUWRCT', 4 BYTES.)
Number of journal output requests during the user-task.

059 DFHTASK (TYPE-A, 'ICPUINCT', 4 BYTES.)
Number of interval-control START or INITATE requests during the user-task.

060 DFHSYNC (TYPE-A, 'SPSYNCCT', 4 BYTES.)
Number of SYNCPOINT requests during the user-task.

061 DFHSTOR (TYPE-A, 'PAGINCT', 4 BYTES.)
Number of page-in operations while the user-task was dispatched.

063 DFHFILE (TYPE-S, 'FCIOWTT', 8 BYTES.)
Elapsed time for which the user-task waited for file I/O.

064 DFHTASK (TYPE-A, 'TASKFLAG', 4 BYTES.)
"Task-error-flags", a string of 31 bits used for signalling unusual conditions occurring during the user-task:

Bit 0	Reserved.
Bit 1	Detected an attempt either to start a clock that was already running, or to stop one that was not running.
Bits 2-4	Reserved.
Bit 5	Detected a corrupted storage accounting area (SAA).
Bits 6-21	Reserved.
Bit 22	Detected a maximum task (CSAMXTON) condition.
Bit 23	Detected a short-on-storage (CSASOSON) condition.
Bits 24-31	Reserved.

067 DFHTERM (TYPE-A, 'TCMSGIN2', 4 BYTES.)
Number of messages received from the alternate terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

068 DFHTERM (TYPE-A, 'TCMSGOU2', 4 BYTES.)
Number of messages sent to the alternate terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

069 DFHTERM (TYPE-A, 'TCALLOCT', 4 BYTES.)
Number of TCTTE ALLOCATE requests issued by the user-task.

070 DFHFILE (TYPE-A, 'FCAMCT', 4 BYTES.)
Number of times the user-task invoked access-method interfaces to transfer data.

071 DFHPROG (TYPE-C, 'PGMNAME', 8 BYTES.)
The name of the first program invoked at attach-time.

074 DFHTASK (TYPE-A, 'ICV', 4 BYTES.)
ICV-value in units of 1/300 of a second at the end of the measurement interval (CSASBTI).

076 DFHINIT (TYPE-C, 'SIT', 4 BYTES.)
Suffix of the system initialization table.

077 DFHTASK (TYPE-P, 'MXT', 4 BYTES.)
MXT-value at the end of the measurement interval (CSAKCMT).

078 DFHTASK (TYPE-A, 'AMXT', 4 BYTES.)
AMXT-value at the end of the measurement interval (CSAMAXT).

079 DFHTASK (TYPE-A, 'ICVTSD', 4 BYTES.)
Terminal scan delay value in units of 1/300 of a second at the end of the measurement interval (CSATSDTI).

080 DFHTASK (TYPE-P, 'TASKCT', 4 BYTES.)
Number of tasks currently in the system at the end of the measurement interval (CSAKCCT).

081 DFHTASK (TYPE-S, 'KCCPUT', 8 BYTES.)
CPU-time for which CICS task-manager was dispatched during the measurement interval.

083 DFHTERM (TYPE-A, 'TCCHRIN1', 4 BYTES.)
Number of characters received from the principal terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

084 DFHTERM (TYPE-A, 'TCCHROU1', 4 BYTES.)
Number of characters sent to the principal terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

085 DFHTERM (TYPE-A, 'TCCHRIN2', 4 BYTES.)
Number of characters received from the alternate terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

086 DFHTERM (TYPE-A, 'TCCHROU2', 4 BYTES.)
Number of characters sent to the alternate terminal facility by the user-task. *(Not applicable to ISC LUTYPE6.2 transactions.)*

087 DFHSTOR (TYPE-A, 'PCSTGHWM', 4 BYTES.)
Maximum amount (high-water mark) of program storage in use by the user-task. The level of program storage in use is incremented, at LINK/XCTL events, by the size in bytes of the referenced program, and decremented at RETURN events.

088 DFHSTOR (TYPE-A, 'PCCOMPRS', 4 BYTES.)
Total number of bytes of program storage that was released by FREEMAIN during the measurement interval.

089 DFHCICS (TYPE-C, 'USERID', 8 BYTES.)
User identification.

090 DFHMAPP (TYPE-A, 'BMSTOTCT', 4 BYTES.)
Total number of BMS requests issued by the user-task.

091 DFHDEST (TYPE-A, 'TDTOTCT', 4 BYTES.)
Total number of transient data requests issued by the user-task.

092 DFHTEMP (TYPE-A, 'TSTOTCT', 4 BYTES.)
Total number of temporary storage requests issued by the user-task.

093 DFHFILE (TYPE-A, 'FCTOTCT', 4 BYTES.)
Total number of file requests issued by the user-task.

094 DFHSTOR (TYPE-S, 'PCCMPRTM', 8 BYTES.)
Elapsed time which the user-task has spent in program compression.

095 DFHSTOR (TYPE-A, 'SCUSRSTG', 8 BYTES.)
Storage "occupancy" of the user-task. This measures the area under the curve of storage-in-use against elapsed-time. The unit of measure is the "byte-unit", where the "unit" is equal to 1024 microseconds. A user occupying, for example, 256 bytes for 125 milliseconds (122 "units") incurs 31232 byte-units of this statistic.

096 DFHTASK (TYPE-S, 'ADSPTIME', 8 BYTES.)
CPU-time spent in the CICS address-space during the measurement interval. Instead of the actual number of times the address-space was entered, the entry-count portion of this clock always contains a value of 1. This is because, in order to reduce overhead, the operating system timer for the address-space is interrogated only the end of the measurement interval.

097 DFHTASK (TYPE-C, 'NETNAME', 20 BYTES.)
Fully qualified name by which the originating system is known to the VTAM network *(network.applid)*. This name is assigned at attach-time using either the net-name derived from the TCT (when the task is attached to a local terminal), or the net-name passed as part of an ISC LUTYPE6.2 or IRC attach-header. At least three padding bytes (X'00') will be present at the right end of the name. When the originator is communicating over a DL/I batch session, the name is a concatenation of 'jobname.stepname.procname' derived from the originating system. Characters in excess of 17 will cause truncation at the **left**.

This field will be blank for entry-point attached system tasks (CSSY, CSLG, and CSNE).

098 DFHTASK (TYPE-C, 'UOWID', 8 BYTES.)
Name by which the unit-of-work is known within the originating system. This name is assigned at attach-time using either a STCK-derived token (when the task is attached to a local terminal), or the UOW-ID passed as part of an ISC LUTYPE6.2 or IRC attach-header. The first six bytes of this field are either:

1. A binary value derived from the clock of the originating system and wrapping round at intervals of several months,

   OR

2. A character-value of the form 'hhmmss', which wraps round daily. This case applies when the originating system is communicating through a DL/I batch session.

The last two bytes of this field may change during the life of the TCA as a result of syncpoint activity.

When using MRO or ISC, the UOWID field must be combined with 'NETNAME' (field 097) to identify a task uniquely, because 'UOWID' is unique only to the originating CICS system.

099 DFHPROG (TYPE-C, 'ABCODE', 8 BYTES.)
Up to two abend-codes experienced by the user-task. If the field is empty (8X'00'), then there was no abend. Otherwise, the first abend-code is recorded, and, later on, the most recent abend-code that is different from the first.

100 DFHTERM (TYPE-S, 'IRIOWTT', 8 BYTES.)
Elapsed time for which the user task waited for a response on an IRC link during the measurement interval.

Exception class

001 DFHTEMP (TYPE-A, 'TSWTSTG', 4 BYTES.)
Amount (in bytes) of temporary storage that was acquired for a PUT request only after waiting.

003 DFHSTOR (TYPE-A, 'SCWTSTG', 4 BYTES.)
Amount (in bytes) of main storage that was acquired by GETMAIN only after waiting.

005 DFHFILE (TYPE-C, 'FCVSSWNM', 8 BYTES.)
Name of a file that incurred a wait for a VSAM string.

009 DFHTASK (TYPE-P, 'TASKNO', 4 BYTES.)
User-task sequence number.

010 DFHCICS (TYPE-A, 'MNEXCNO', 4 BYTES.)
Sequence number of this exception within the user-task.

011 DFHTASK (TYPE-C, 'TRAN', 4 BYTES.)
Transaction identification.

012 DFHTERM (TYPE-C, 'TERM', 4 BYTES.)
Terminal identification.

013 DFHCICS (TYPE-C, 'OPR', 4 BYTES.)
Operator identification.

014 DFHTASK (TYPE-C, 'T', 4 BYTES.)
Transaction type:

A	Attached by automatic transaction initiation (ATI)
C	Second or subsequent part of a conversational task
D	Attached by transient-data trigger-level
S	System internal task
T	Attached from a terminal (but see 'Z' below)
Z	Second or subsequent part of a pseudoconversation.

015 DFHCICS (TYPE-T, 'START', 8 BYTES.)
Start-time of exception condition.

016 DFHCICS (TYPE-T, 'STOP', 8 BYTES.)
Finish-time of exception condition.

017 DFHCICS (TYPE-C, 'USERID', 8 BYTES.)
User identification.

018 DFHSTOR (TYPE-S, 'PCOMPRTM', 8 BYTES.)
CPU-time used by a single operation of the program compression mechanism (but see the note to "Exception class data" on page 390).

019 DFHFILE (TYPE-C, 'FCVSBWNM', 8 BYTES.)
Name of a file that incurred a wait for a VSAM buffer.

020 DFHSTOR (TYPE-S, 'SCWTETIM', 8 BYTES.)
The elapsed time used during a suspend because an unconditional storage request could not be satisfied. (See also the notes about exception class data on page 390.)

User exits for accessing monitoring data

Two task-related user exits in the CICS monitoring facility enable you to access monitoring data records that are complete. Exit processing cannot change the record that will appear in the monitoring data set.

When you initialize your CICS system, you can specify, in your DFHSIT macro definition, whether you want monitoring facilities. Alternatively, you can request such facilities dynamically, using the CSTT MONITOR transaction, or by initialization overrides.

Exits in the CICS monitoring program, DFHCMP, give you access to transaction records of all three classes, and to global performance records.

Note: Do not use EDF for task-related user exits within monitoring. Programs for task-related user exits within monitoring should be compiled with the NOEDF option.

Transaction records

CICS builds monitoring records for each monitoring class that is active. When a record is completed, CICS transfers it to a buffer. The user exit gives you access to each record after it has completed, but before it has been copied to the buffer.

This means that the monitoring facility passes control to an exit program you have written, at the following times for each class:

Accounting Class Just before the end of the task

Performance Class

- Just before the end of each task.
- If CONV=YES is specified on the DFHMCT TYPE=RECORD macro, each time a CONVERSE command (or a SEND and RECEIVE pair) is issued.
- If the DELIVER option is specified in an MCT user-EMP, each time that option is invoked.

Exception Class At the end of each exceptional condition.

The exit program is invoked separately for each record.

Global records

The monitoring control table (MCT) contains a time interval specified in the FREQ operand of the DFHMCT TYPE=RECORD, CLASS=PERFORM resource definition macro. DFHCMP passes control to its exit whenever it completes a global performance record, which it does each time the MCT interval expires, and when the performance class is being deactivated.

Parameter lists

On entry to the exit program, register 1 points to a saved parameter list. You can use the DSECT DFHUEPAR to address this list (see "User exit parameter lists" on page 348). The following description should be read in conjunction with Figure 36 on page 418.

UEPEXN
Gives the address of the function identifier, a fullword that identifies the source of the call. In this exit, this field should always contain X'0006', denoting a task-related user exit call from the monitoring program.

UEPHMSA
Gives the address of the monitoring program's register save area. Register 1, saved at offset X'18' of this area, gives the address of the monitoring exit parameter list. The first fullword of this list is the address of the exit identifier, a one-byte field containing:

X'80'	For transaction records
X'40'	For global records.

Figure 37 on page 419 illustrates the parameter list for the transaction record exit. Note that only one of the three dictionary header addresses is valid — you can determine which by the value of the record type field as shown in the figure.

Figure 38 on page 420 illustrates the parameter list for the global performance record exit.

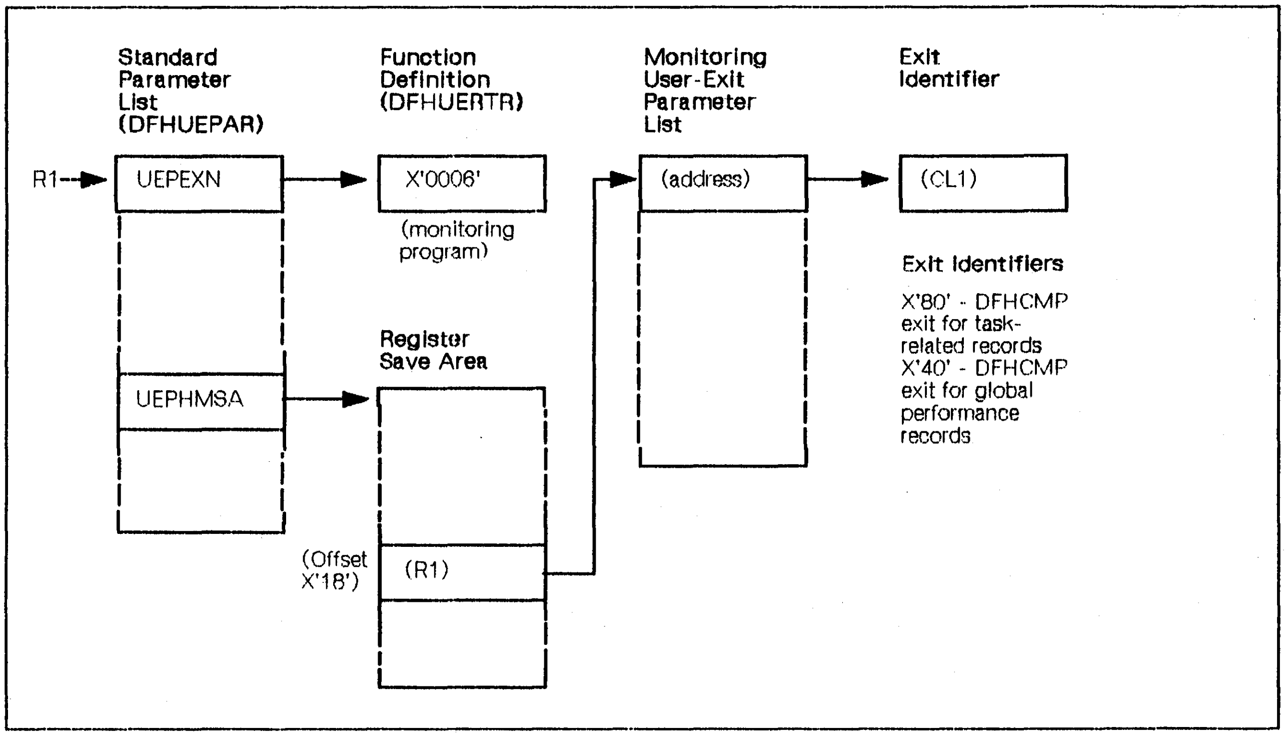
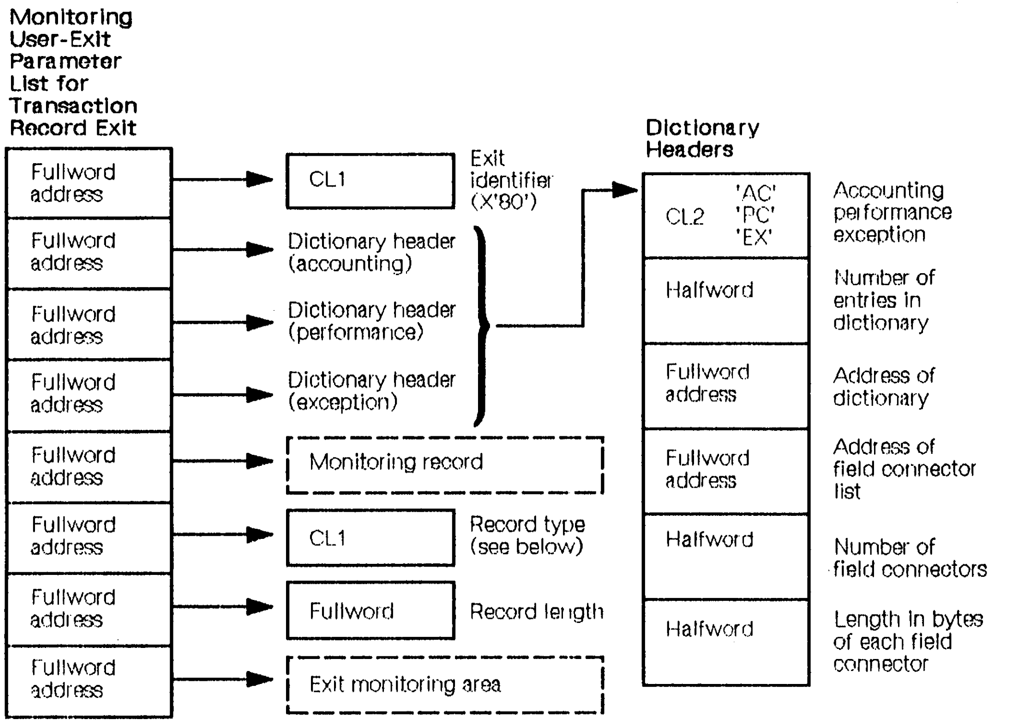

Standard Parameter List (DFHUEPAR)
Function Definition (DFHUERTR)
Monitoring User-Exit Parameter List
Exit Identifier
R1
UEPEXN
X'0006'
(monitoring program)
(address)
(CL1)
Exit Identifiers
X'80' - DFHCMP exit for task-related records
X'40' - DFHCMP exit for global performance records
Register Save Area
UEPHMSA
(Offset X'18')
(R1)

Figure 36. Parameter lists for CICS monitoring exits

Note: The form of the monitoring user-exit parameter list depends on the reason the exit was invoked:

1. If the exit was invoked for gathering transaction-related data, the parameter list is as shown in Figure 37 on page 419.
2. If the exit was invoked for gathering global performance data, the parameter list is as shown in Figure 38 on page 420.



Record Types

X'01' - Accounting
X'02' - Performance (end of task)
X'03' - Performance (other than end of task)
X'04' - Exception

Use the record type field to find which dictionary header to use.

Figure 37. Parameter lists for transaction user exit

For the layout of the Exit monitoring area, see Figure 39 on page 421.

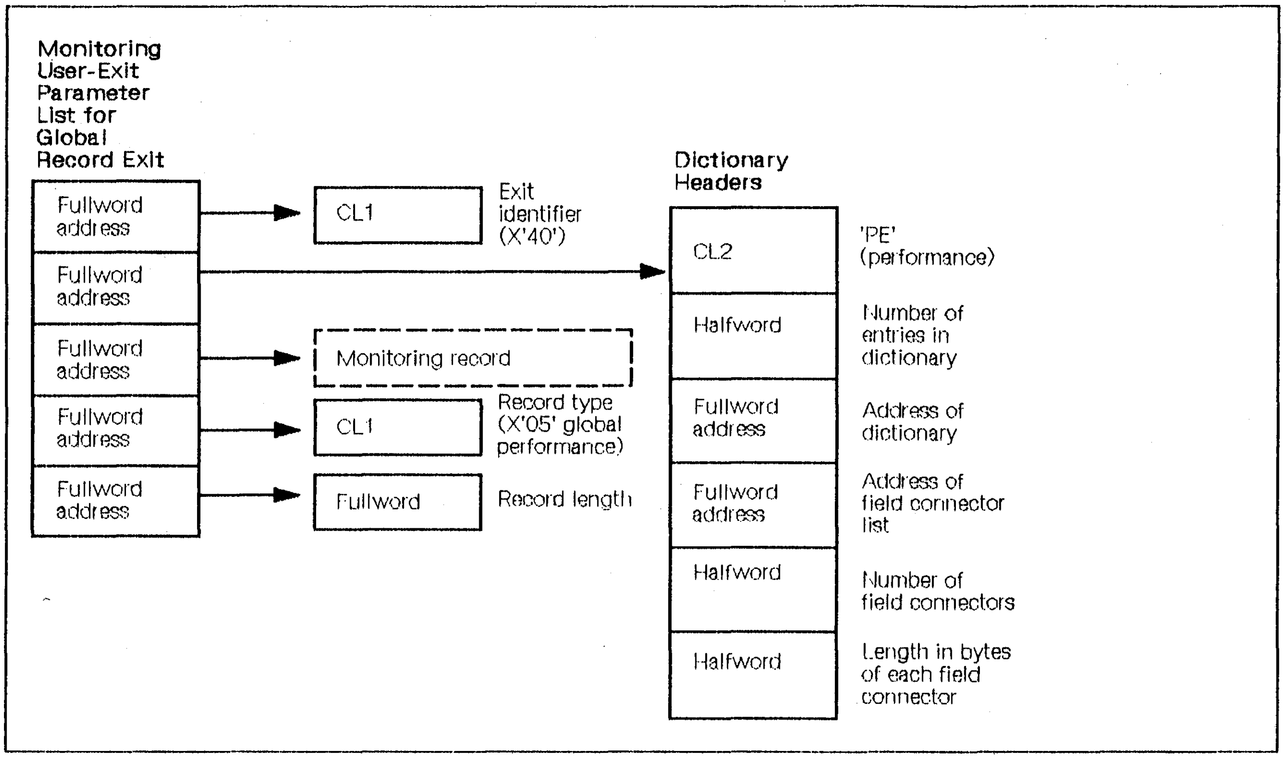


Figure 38. Parameter lists for global performance user exit

How to install and use a monitoring exit program

When you have written the exit program to handle the monitoring records, you must perform the following steps before you can use it:

1. Assemble your program and store it in the load library. You must specify the translator option NOEDF, because the command-level execution diagnostic facility (EDF) cannot be used with exit programs invoked by CICS Monitoring.

2. Create a processing program table (PPT) entry using the online resource definition transaction (CEDA) or the DFHPPT macro instruction. (See the *CICS/MVS Resource Definition (Macro)* manual.)

3. Code a DFHMCT TYPE=RECORD macro instruction, specifying EXIT=YES, for each class of monitoring you want to use.

4. Bring up your CICS system. If you want monitoring facilities from the time of initialization, you should code the MONITOR operand of DFHSIT for each monitoring class you need. Otherwise, you can wait until CICS is running, and use the CSTT MON request to activate certain monitoring classes. Whichever method you choose will start CICS collecting relevant monitoring data.

5. Enable the exit program. You can do this by executing a transaction to invoke a program that contains a statement of the form:

   ```
   EXEC CICS ENABLE PROGRAM (name) START TASKSTART
   ```

Note: If you specify the START and TASKSTART options in separate ENABLE commands, the START option should occur on the command that is executed first. Otherwise, transaction abend ASPT will occur, because the syncpoint manager will diagnose damage in the adapter which was not started. No damage may exist, but the sync point manager must react as if it does and must "fail safe". If you want monitoring exits active from initialization, however, you should code a program list table (DFHPLT) entry naming the program that contains the above command.

Monitoring during the exit

Be aware that to maintain consistency between the monitoring information accessed online and that written to the journal, CICS will not include in the monitoring records presented to the exit any activity subsequently performed by the exit program itself. In effect, monitoring presents to the exit a record in an output buffer, while it continues to collect data in its own areas. For transaction exits only, the address of these is passed to the exit program. The exit program can access them as the "exit monitoring area" referred to in Figure 37 on page 419 and described as follows.

Each class of monitoring data occupies a different part of the exit monitoring area. You locate the area for a particular class by reading its offset from a halfword field at the start of the exit monitoring area. The halfword fields for the three classes are in the following offsets from the start of area.

MONITORING CLASS	OFFSET FROM START OF AREA OF HALFWORD FIELD
ACCOUNTING	2
PERFORMANCE	4
EXCEPTION	6

Figure 39. Monitoring exit area — location of address fields

You can map the area for each class by using the dictionary presented elsewhere in the exit interface. The accounting area will be the same as that presented as the monitoring record. The exception area will be cleared initially, and will remain cleared unless the exit program itself incurs an exception.

Collecting task-throughput data using RMF and SMF

If you code EVENT=YES on the DFHMCT TYPE=INITIAL macro, CICS will use the MVS SYSEVENT macro to pass each task's timing information to the resource measurement facility.

RMF must be active in the MVS region. RMF can provide reports or write records to the SMF data sets.

For more information, see *OS/VS2 MVS Resource Measurement Facility, Version 2 — Reference and Users Guide*, SC28-0922, and *OS/VS2 MVS System Programming Library: System Management Facilities (SMF)*, GC28-1030.

Chapter 5.9. Examining and modifying resource attributes

General-Use Programming Interface

The CICS command-level interface provides commands that enable application programs to examine and modify attributes of the following resources:

- Files
- Terminals
- System entries (connections) in the terminal control table
- Mode names for a particular system connection
- The system
- Programs
- Transactions.

Although the commands have functions that will be used in application programs, they are mainly for system programming purposes.

They are of particular benefit to developers of products that run on CICS together with customer applications that perform operational or system programming functions. They allow more products to run above the 16MB line in an XA environment, by relieving the dependency on CICS macros that are restricted to 24-bit execution.

Compared with the macro equivalents, these functions have the advantages and characteristics of the CICS command-level programming interface, and are supported by the command interpreter, the translator, and by EDF. They improve security, integrity, installation and usability.

The following are summaries of the functions:

- INQUIRE functions

  These functions provide application programs with controlled access through the CICS command-level programming interface to information concerning CICS files, terminals, system connections, modenames, system attributes, transactions, and programs.

- SET functions

  The application program can alter the value or status of many attributes of the facilities listed under INQUIRE.

- ASSIGN and ADDRESS functions.

Note: For full information about the use of EXEC CICS commands and about the ASSIGN and ADDRESS commands, see the *CICS/MVS Application Programmer's Reference* manual.

CICS-value data-areas

You will see the term "cvda" used in the syntax boxes describing the INQUIRE and SET functions. (In the text "cvda" is capitalized – CVDA.)

The notation CVDA (CICS-value data-area) indicates a fullword binary user-field, which may have assigned into it one of a set of CICS-defined values corresponding to various attributes and states. The option list associated with a syntax box indicates the possible values for each CVDA. It is used by, and in connection with, INQUIRE and SET commands. With INQUIRE commands CICS will assign values into your CVDAs. You may test them symbolically using the symbolic name DFHVALUE. With SET commands you may assign these values symbolically using the same symbolic name, and CICS will act on them.

These symbolic names will be translated into binary values by the CICS translator. In PL/I and COBOL, the translator will replace DFHVALUE(...) with EBCDIC characters representing the CVDA's binary number, n. In assembler applications, it will be replaced by the literal construct =F'n'.

Under CEDF and CECI, the screen will not display the symbolic names, but only the corresponding binary numbers.

The values used are given in "CICS-VALUES used in INQUIRE and SET commands" on page 461.

The INQUIRE and SET commands—points to note

This section has three lists dealing with the interface in general, the INQUIRE command, and the SET command, respectively.

The INQUIRE/SET interface

- To complement this function, additions are provided in the ASSIGN command. For information about this command, see the *CICS/MVS Application Programmer's Reference* manual.
- The INQUIRE/SET function is not supported across ISC or MRO.
- There is no exclusive control maintained by CICS on the information requested. and it may therefore change at any time subsequent to the request (for example, as the result of CICS or other user activity).
- Like all other EXEC CICS commands, these commands have NOHANDLE, NOEDF and RESP options. They are not shown in the syntax in this chapter. For more information about them, see the *CICS/MVS Application Programmer's Reference* manual.
- As well as the RESP option, these commands have a RESP2 option. The overall response to the execution of a command is shown by setting a value in the RESP(xxx) and RESP2(yyy) user-fields, or by raising a HANDLEable condition.

The occurrence of an exceptional condition may be detected by testing the value in the RESP field. The RESP and RESP2 values that could arise are listed after the details of each command. A CICS built-in function is provided to facilitate RESP testing. In PL/I this test might be:

```
IF xxx¬=DFHRESP(NORMAL) THEN....
```

Then testing may be performed for any particular suspected condition:

```
IF xxx=DFHRESP(FILENOTFOUND) THEN....
```

In assembler this test might be:

```
C     xxx,DFHRESP(FILENOTFOUND)
BE    label
```

RESP2 contains a fullword binary value, and you test this value, not a symbolic name.

RESP and RESP2 may be viewed in the following ways:

- RESP — on the CEDF command execution complete screen,
- RESP and RESP2 — CEDF program initialization,
- RESP — CECI execution, complete screen,
- RESP and RESP2 — CECI, using PF4 (or equivalent), following command execution.

EXEC CICS INQUIRE command

- Any INQUIRE requesting attributes that the facility does not possess, or which for some reason cannot be determined, will result in a "null" setting for the user variable specified to receive the attribute. These values are designed to make it possible to distinguish "non-applicable" values from real returned values and are defined as follows:
  - Character fields are blanks
  - Binary fields are −1
  - Pointer fields are X'FF000000'
  - CVDA fields are DFHVALUE(NOTAPPLIC), which has a value of 1.
- If a condition occurs in any INQUIRE command, all the specified user variables will have undefined values on return.
- The END condition is raised on all INQUIRE NEXT commands when there are no more entries of the required resource type.

EXEC CICS SET command

- The SET commands enable application programs to modify CICS resources. In general, the items that can be modified are a subset of those that can be retrieved by the INQUIRE command.
- In general, the effect of a particular operation specified on an EXEC CICS SET command is the same as when it is specified through CEMT.
- If a SET command includes "null" argument values, the corresponding option will be ignored. This allows the possibility of coding general SET commands in which some attributes may be left as they are, without having to issue an INQUIRE command first to establish what the current values are. For CVDAs,

the null value may be called DFHVALUE(IGNORE), which has a value of 1. The SET command can also be coded like this:

```
EXEC CICS SET FILE(FILENAME) READABLE DELETE(1)
ADDABLE END-EXEC
```

You can then read and add to the file, but the delete status is the same as it was before the SET command, because 1 is the actual value of IGNORE.

- There are two ways of using SET commands with CVDAs.

  Flexible form
  You may assign a CVDA value to a variable:

  ```
  xxxx = DFHVALUE(DELETABLE)
  ```

  That variable may then be used in the place of a CVDA value, and in this case the file will be DELETABLE:

  ```
  EXEC CICS SET FILE(name of file)
  DELETE(xxxx)
  ```

  Thus, the delete status of the file may be set on or off depending upon branches in the code before the EXEC CICS command.

  If you want to leave the delete status unchanged, without knowing what it is, you can code:

  ```
  xxxx=DFHVALUE(IGNORE)
  ```

  Short form
  If the CVDA value is always the same for a particular command, you can declare the value directly:

  ```
  EXEC CICS SET FILE(name of file) DELETABLE
  ```

  If you want to leave the delete status unchanged, without knowing what it is, you can simply omit the DELETE option from the SET command.

- If a condition occurs in any SET command, as few as possible of the requested changes will have been made on return. See under individual SET commands for details.

- A SET command issued against an individual resource will result in a resource-level security check, if RSLC=YES has been specified for the transaction issuing the command.

- The NOTAUTH condition is raised on all SET requests where the application is not authorized to view or alter the resource.

- To prevent misuse of the EXEC CICS SET commands, the CICS modules that implement them have been made subject to security checking. To implement security checking for EXEC CICS SET commands, you should:

  1. Make individual PPT entries for the DFHEIQxx modules, rather than use the Group entry INQUIRESET.
  2. Specify RSLC=YES (or external security) on each entry.
  3. Give those system programmers who need to use the EXEC CICS SET commands an RSLKEY that matches this protection value, or the equivalent external security level.

4. Ensure that the CECI transaction retains the RSLC value YES, which is its default, or that external security is specified for it.

With this protection installed, any unauthorized attempt to execute a SET command will be prevented, and the NOTAUTH exceptional condition will be raised.

Examples of EXEC CICS INQUIRE/SET

Here are three examples, in assembler, PL/I and COBOL, showing INQUIRE and SET commands on a file, using DFHVALUE as a symbolic reference to the CVDA value. Between the two EXEC CICS commands are operations upon those values. After each example, there follows the translation of the central portion of that code, showing the translation of the symbol into a value.

Assembler version

```
         DFHEISTG
         .
         .               *USER DEFINED VARIABLES FOR:
UOPST    DS    F         *OPEN STATUS
UENST    DS    F         *ENABLE STATUS
UREAD    DS    F         *READ STATUS
UUPD     DS    F         *UPDATE STATUS
INFILE   DS    CL8       *FILE NAME
         .
         .

SYNTAXD  CSECT
   .
   .
         MVC INFILE(8),=C'PAYROLLb'
         EXEC CICS INQUIRE FILE(INFILE)
                   OPENSTATUS(UOPST)
                   ENABLESTATUS(UENST)
*
*- - - - - Translated section (see below)  - - - - - -
*
         CLC   UOPST,DFHVALUE(OPEN)         *IS FILE OPEN?
         BE    OPENLAB                      *YES, BYPASS SETTING SERVREQS
*
         MVC   UREAD,DFHVALUE(READABLE)
         MVC   UUPD,DFHVALUE(NOTUPDATABLE)
*
*- - - - - End of translated section - - - - - - - - -
*
         EXEC CICS SET FILE(INFILE)
                   READ(UREAD)
                   UPDATE(UUPD)
                   NOTADDABLE
                   NOTDELETABLE
*
OPENLAB  DS    0H
*
```

The central section translates as follows:

```
*        CLC   UOPST,DFHVALUE(OPEN)
         CLC   UOPST,=F'18'
         BE    OPENLAB
*
*        MVC   UREAD,DFHVALUE(READABLE)
         MVC   UREAD,=F'35'
*        MVC   UUPD,DFHVALUE(NOTUPDATABLE)
         MVC   UUPD,=F'38'
*
```

PL/I version

```
 DCL
 .
 .                    /*USER DEFINED VARIABLES FOR: */
 (UOPST,              /*OPEN STATUS                 */
 UENST,               /*ENABLE STATUS               */
 UREAD,               /*READ STATUS                 */
 UUPD)FIXED BIN(31),  /*UPDATE STATUS               */
 INFILE CHAR(8);      /*FILE NAME                   */
 .
 .
         INFILE='PAYROLLb';
         EXEC CICS INQUIRE FILE(INFILE)
                   OPENSTATUS(UOPST)
                   ENABLESTATUS(UENST);
 /**/
 /**/- - - Translated section (see below)  - - - - - -
 /**/
       IF UOPST = DFHVALUE(CLOSED) THEN
       DO;
         UREAD= DFHVALUE(READABLE);
         UUPD = DFHVALUE(NOTUPDATABLE);
 /**/
 /**/- - - End of translated section - - - - - - - - -
 /**/
         EXEC CICS SET FILE(INFILE)
                   READ(UREAD)
                   UPDATE(UUPD)
                   NOTADDABLE
                   NOTDELETABLE;
 /**/
       END;
```

The central section translates as follows:

```
 /**/
       IF UOPST =              19   THEN
       DO;
         UREAD=              35    ;
         UUPD =              38         ;
 /**/
```

COBOL version

```
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01  MESSAGES.
      .
      .
      *                           USER DEFINED VARIABLES FOR
      *                           OPEN STATUS, ENABLE STATUS,
      *                           READ STATUS, UPDATE STATUS,
      *                           AND FILE NAME
      01  UOPST                   PIC S9(8) COMP.
      01  UENST                   PIC S9(8) COMP.
      01  UREAD                   PIC S9(8) COMP.
      01  UUPD                    PIC S9(8) COMP.
      01  INFILE                  PIC X(8)
      .
      .
      CICS-REQUESTS.
        MOVE 'PAYROLLb' TO INFILE.
        EXEC CICS INQUIRE FILE(INFILE)
                  OPENSTATUS(UOPST)
                  ENABLESTATUS(UENST)
                  END-EXEC.
*
*- - - - - Translated section (see below)  - - - - - -
*
        IF UOPST NOT = DFHVALUE(CLOSED)
           THEN GOBACK.
*
        MOVE DFHVALUE(READABLE)      TO UREAD.
        MOVE DFHVALUE(NOTUPDATABLE) TO UUPD.
*
*- - - - - End of translated section - - - - - - - - -
*
        EXEC CICS SET FILE(INFILE)
                  READ(UREAD)
                  UPDATE(UUPD)
                  NOTADDABLE
                  NOTDELETABLE
                  END-EXEC.
*
```

The central section translates as follows:

```
        IF UOPST NOT =              19
           THEN GOBACK.
*
        MOVE             35          TO UREAD.
        MOVE             38          TO UUPD.
*
```

INQUIRE and SET commands

The specific information you can request is indicated in the syntax outlined below, and is roughly equivalent to the information available through CEMT, but considerably less than can be specified through CEDA. Keywords are used to specify the information required for particular resources.

A program can scan through all files, terminals, system connections, modenames, programs or transactions defined to the CICS system.

INQUIRE for files

The INQUIRE command for CICS files returns named items of information about any file defined in the file control table. The function is supported for VSAM, BDAM, and remote files, but not for DL/I data sets. An INQUIRE for a file requesting attributes that the file may not possess results in a special null setting (see page 425) for the user variable specified to receive the attribute.

```
EXEC CICS INQUIRE
     FILE(8-character data-value)
     [ACCESSMETHOD(cvda)]
     [TYPE(cvda)]
     [OBJECT(cvda)]
     [DSNAME(44-character data-area)]
     [BASEDSNAME(44-character data-area)]
     [REMOTESYSTEM(4-character data-area)]
     [REMOTENAME(8-character data-area)]
     [RECORDFORMAT(cvda)]
     [BLOCKFORMAT(cvda)]
     [KEYLENGTH(fullword binary data-area)]
     [KEYPOSITION(fullword binary data-area)]
     [RECORDSIZE(fullword binary data-area)]
     [BLOCKSIZE(fullword binary data-area)]
     [STRINGS(fullword binary data-area)]
     [LSRPOOLID(fullword binary data-area)]
     [OPENSTATUS(cvda)]
     [ENABLESTATUS(cvda)]
     [DISPOSITION(cvda)]
     [RECOVSTATUS(cvda)]
     [EMPTYSTATUS(cvda)]
     [READ(cvda)]
     [UPDATE(cvda)]
     [BROWSE(cvda)]
     [ADD(cvda)]
     [DELETE(cvda)]
     [RELTYPE(cvda)]
     [EXCLUSIVE(cvda)]
     [BLOCKKEYLEN(fullword binary data-area)]
```

You should make sure that you issue the INQUIRE command at the appropriate time, taking into account the state of the file, because most of the options, at some time or another, are not appropriate for a particular file. For many of the options, if the file is closed at the time of inquiry, the information you receive tells you what the state of the file will be when it is next opened. If the file has never been opened, for some of the options you will receive default values, which could change when the file is opened. You should consider whether DFHFCT has defined values for a file. Also, you should issue commands for the correct type of file. For example, RELTYPE is inappropriate when the file has an ACCESSMETHOD of VSAM. In such cases, a null value will be returned in the user variable. This means NOTAPPLIC for CVDA fields.

Options

Fuller details of these options are given under their equivalent names in the DFHFCT section of the *CICS/MVS Resource Definition (Macro)* manual.

ACCESSMETHOD
Possible CVDA values are BDAM, VSAM, or REMOTE, indicating the global access method as recorded in the FCT.

BASEDSNAME (VSAM only)
The name of the VSAM base data set if the object is defined to CICS as a path. The name is the same as that in DSNAME if the file is a base. BASENAME is accepted in place of the keyword BASEDSNAME.

BLOCKFORMAT
Possible CVDA options are BLOCKED and UNBLOCKED, as given in the RECFORM option in the DFHFCT macro.

BLOCKKEYLEN (BDAM only)
Current value as recorded in the FCT.

BLOCKSIZE (BDAM only)
Current value as recorded in the FCT.

DISPOSITION
Possible CVDA values are OLD and SHARE, which indicate how the JCL has specified the data set to be treated when a file that references it is next opened.

DSNAME
The data set name as defined to the access method and operating system. If no JCL statement exists for this file when it is opened, the open will be preceded by a dynamic allocation of the file using this data set name. If there is a JCL statement, it will take precedence over this data set name. OBJECTNAME is accepted in place of the keyword DSNAME.

EMPTYSTATUS
Possible CVDA values are EMPTYREQ and NOEMPTYREQ. EMPTYREQ means the data set will be made empty when a file that references it is next opened. NOTAPPLIC means the data set has not been defined as reusable.

ENABLESTATUS
Possible CVDA options are ENABLED, DISABLED, UNENABLED, or DISABLING. A file may be:

ENABLED
The file is available for access by application programs.

DISABLED
The file is unavailable for access by application programs. The file can be re-enabled by SET ENABLED.

UNENABLED
This is the same as DISABLED, except that it occurred implicitly when SET CLOSED was requested. However, in this case, the file will be enabled implicitly by a SET OPEN.

DISABLING
The file is still being accessed by application programs after a SET DISABLED (or CLOSED) command has been received. No new application programs will be allowed to access the file. Current applications will be allowed to complete their use of it.

EXCLUSIVE (BDAM only)
Possible CVDA values are EXCTL and NOEXCTL. Records are normally subject to exclusive control.

FILE (sender)
The file name as defined in the FCT. The keyword DATASET is accepted in place of FILE. FILE is the preferred term.

KEYLENGTH
Current value as recorded in the FCT.

KEYPOSITION
Current value as recorded in the FCT.

LSRPOOLID (VSAM only)
If the file may use a shared buffer pool, this is its number. If the file is not VSAM, this value will be −1. If the file is not to share buffers, this value will be 0.

OBJECT (VSAM only)
Possible CVDA values are BASE, PATH, or NOTAPPLIC, which indicate whether this file is related to a base data set, or is defined as a path. A path may simply be an alias for a base data set, or may refer to a data set accessed by means of an alternate index. An alternate index structure (path) opened as a data set (which is known to CICS only as a base) returns the value BASE.

OPENSTATUS
Possible CVDA values are OPEN, CLOSED, OPENING, CLOSING, and CLOSEREQUEST. This attribute describes the state of the file as it appears to the operating system, and does not determine whether the file is available for access by application programs. A file may be:

OPEN
The file has been opened explicitly (by EXEC CICS SET OPEN or by CEMT) or implicitly on first access by an application program.

CLOSED
The file has not yet been opened, or has been closed explicitly.

OPENING
The file is in the process of being opened. Note that this may involve other activity, such as the building of a shared resources buffer pool.

CLOSING
The file is in the process of being closed. Note that this may involve other activity, such as dynamic deallocation or the deletion of shared resources.

CLOSEREQUEST
The file is open and in use by application program(s), and an EXEC CICS SET CLOSED or CEMT request has been received. The file has entered the DISABLING state. When this state changes to DISABLED, the file is closed.

READ
Possible CVDA values are READABLE and NOTREADABLE.

UPDATE
Possible CVDA values are UPDATABLE and NOTUPDATABLE.

BROWSE
Possible CVDA values are BROWSABLE and NOTBROWSABLE.

ADD
Possible CVDA values are ADDABLE and NOTADDABLE.

DELETE
Possible CVDA values are DELETABLE and NOTDELETABLE.

The above five options indicate whether applications are allowed to perform the particular type of operation on the file. UPDATABLE means the READ-REWRITE or the READ-DELETE sequence. DELETABLE covers both the DELETE and the READ-DELETE sequence. ADDABLE means that new records can be added to the file. Note that READ is always available with UPDATE and BROWSE.

RECORDFORMAT
Possible CVDA values are FIXED, VARIABLE, and UNDEFINED, as given in the RECFORM option in the DFHFCT macro. The UNDEFINED value refers to BDAM only.

RECORDSIZE
Current value as recorded in the FCT. When the RECORDFORMAT is VARIABLE, the value of RECORDSIZE is the maximum record length.

RECOVSTATUS
Possible CVDA values are RECOVERABLE and NOTRECOVABLE. A data set is RECOVERABLE when LOG=YES has been coded in the DFHFCT macro. It is NOTRECOVABLE if LOG=NO or SERVREQ=REUSE has been coded.

RELTYPE (BDAM only)
The possible CVDA values are HEX, DEC, and BLK, which further define the type of file, as recorded in the FCT.

REMOTENAME
The name that this file has in the remote system.

REMOTESYSTEM
If the file is remote, the name of the remote CICS system.

STRINGS (VSAM only)
The number of concurrent operations that may be performed on this file.

TYPE
A CVDA value of ESDS, KSDS, RRDS (for VSAM) or KEYED, NOTKEYED (for BDAM) further defines the type of data set that corresponds to this file. For VSAM, this information is obtained from the VSAM catalog when the file is opened.

Exceptional conditions

The following conditions might occur:

FILENOTFOUND
The named file cannot be located. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH The invoker is not authorized to access the file.

Browse operations for files

You may use the INQUIRE command to browse through all file definitions **that you are authorized to access.** To do this, you use the following three commands:

```
EXEC CICS INQUIRE FILE START

EXEC CICS INQUIRE FILE(8-character data-area) NEXT
                              (other options as in
                               the above option list)

EXEC CICS INQUIRE FILE END
```

When the application needs to terminate the browsing, the command INQUIRE FILE END may be coded. This will free any resources the function is using. If the INQUIRE FILE END is not issued, the browse operation is terminated by the end of the transaction; it is not terminated by syncpoints.

INQUIRE FILE START command initializes the FCT scan for a series of INQUIRE NEXT commands. Note that the FILE option does not have an operand. The first file is returned in response to the first INQUIRE FILE NEXT command. This simplifies user programming.

The following condition might occur:

ILLOGIC A browse is already in progress.

INQUIRE FILE NEXT command puts the next file name in the FCT into the area specified by the FILE keyword. A pointer to the current browse position is maintained at transaction level. This is not reset across LINK or XCTL.

Note that **no** facility is provided for selective browsing, that is, for returning entries only if they meet certain conditions.

The following conditions can occur:

ILLOGIC An INQUIRE START has not been issued.

END All of the authorized entries in the FCT have been returned in the browse operation. No file is returned.

The order of the file names browsed during a browse operation is not necessarily the collating sequence. However, continued browsing does guarantee to return all files in the FCT. Only one file browse operation is allowed at any one time within one transaction.

INQUIRE FILE END command ends the browsing operation and frees any resources held.

The following condition can occur:

ILLOGIC An INQUIRE START has not been issued.

SET for files

You can use the EXEC CICS SET FILE command to set or modify attributes of a particular named VSAM or BDAM file. The five forms of the command are given below. Options that are different from those for INQUIRE for files are given after the five syntax boxes.

```
EXEC CICS SET
     FILE(8-character data-value)
     [DSNAME(44-character data-area)]
     [STRINGS(fullword binary data-area)]
     [LSRPOOLID(fullword binary data-area)]
     [EMPTYSTATUS(cvda)]
     [DISPOSITION(cvda)]
     [READ(cvda)]
     [UPDATE(cvda)]
     [BROWSE(cvda)]
     [ADD(cvda)]
     [DELETE(cvda)]
```

Any combination of the above attributes that is appropriate to the type of file (VSAM or BDAM) may be set in one command. All require that the file be in a CLOSED, DISABLED state.

Exceptional conditions

The following conditions can occur:

FILENOTFOUND
The SET command could not locate the named file. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH The invoker is not authorized to access the file.

INVREQ An attempt was made to set an invalid CVDA value, or the file is not in a valid state to allow the requested changes.

The RESP2 field contains further information:

Value	Meaning
1	File is remote
2	File is not closed
3	File is not disabled
4	ADD problem : invalid CVDA value
5	BROWSE problem: invalid CVDA value
7	DELETE problem: invalid CVDA value
8	DISPOSITION problem: invalid CVDA value
9	EMPTYSTATUS problem: invalid CVDA value
10	LSRPOOLID problem: not a VSAM data set
11	LSRPOOLID problem: value out of range (1 through 8), or corresponding buffer not defined
12	READ problem: invalid CVDA value
13	STRINGS problem: value out of range, or not a VSAM data set
14	UPDATE problem: invalid CVDA value, or, if the attempted operation failed in VSAM, the VSAM return code.

The following four commands must be set explicitly in the short form.

SET ENABLED

```
EXEC CICS SET
     FILE(8-character data-value)
     ENABLED
```

Exceptional conditions

The following conditions can occur:

FILENOTFOUND
The SET command could not locate the named file. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH The invoker is not authorized to access the file.

INVREQ The attempted operation failed. Either the file is currently disabling, or the ENABLE command was halted by a user exit program running at exit point XFCSREQ. If an exit program running at this exit point instructed CICS not to execute the ENABLE command, the RESP2 value is 28. Otherwise the RESP2 value is zero.

SET DISABLED

```
EXEC CICS SET
     FILE(8-character data-value)
     DISABLED
     (WAIT|NOWAIT|FORCE)
```

If WAIT|NOWAIT|FORCE is not specified, WAIT is assumed.

Exceptional conditions

The following conditions can occur:

FILENOTFOUND
The SET command could not locate the named file. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH
The invoker is not authorized to access the file.

INVREQ The attempted operation failed. Either a DISABLED WAIT or DISABLED FORCE command has been issued and the caller is itself a user of the file, or the DISABLED command was halted by an exit program running at exit point XCFSREQ. If an exit program running at this exit point instructed CICS not to execute the DISABLED command, the RESP2 value is 28. Otherwise the RESP2 value is zero.

SET OPEN

```
EXEC CICS SET
     FILE(8-character data-value)
     OPEN
     (EMPTY)
```

Exceptional conditions

The following conditions can occur:

FILENOTFOUND
The SET command could not locate the named file. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH The invoker is not authorized to access the file.

INVREQ An exit program running at exit point XFCSREQ instructed CICS not to execute the OPEN command. RESP2 is set to a value of 28.

IOERR The OPEN failed, perhaps because the data set has not been defined as REUSE. The VSAM return code, if any, is set in RESP2.

SET CLOSED

```
EXEC CICS SET
     FILE(8-character data-value)
     CLOSED
     (EMPTY)
     (WAIT|NOWAIT|FORCE)
```

Exceptional conditions

The following conditions can occur:

FILENOTFOUND
The SET command could not locate the named file. The exceptional condition DSIDERR is equivalent to FILENOTFOUND.

NOTAUTH The invoker is not authorized to access the file.

| **INVREQ** The CLOSE was not attempted. Either the caller was using the file or an exit program running at exit point XFCSREQ halted the CLOSE command. If an exit program running at this exit point instructed CICS not to execute the CLOSE command, RESP2 is 28. Otherwise RESP2 is zero.

| **IOERR** The CLOSE failed. Any VSAM return code is set in RESP2, otherwise this field is set to zero.

Options

See under "INQUIRE for files" on page 430 for meanings of options not listed below.

EMPTY
If used when opening a file, this results in an empty data set. If used when closing a file, it sets a request that states that the data is no longer required. So, when that file is reopened, it will be emptied. You can only use EMPTY if the data set has been specified to VSAM AMS as REUSE. If REUSE has not been specified, the OPEN fails.

LSRPOOLID
To prevent a file sharing buffers, set this value to 0.

WAIT|NOWAIT|FORCE
If you do not specify one of these, WAIT is assumed.

WAIT
CICS will wait until all activity of the file has quiesced, and will then perform the action before returning control to the application.

NOWAIT
CICS will wait until all activity of the file has quiesced before performing the action, but will return control to the application as soon as the request has been queued.

FORCE
This option will abend any tasks currently using the file, will perform the action immediately, and then return.

Notes:

1. Any option with a null value will be ignored.
2. SET may be used at any time with or without an INQUIRE having been issued previously.
3. The INQUIRE and SET commands do not operate on DL/I databases. If you want to operate on DL/I, use the CEMT command. For details, see the *CICS/MVS CICS-Supplied Transactions* manual.
4. Some of the SET commands do not take effect until the file is opened. To be certain of the file attributes, you must open the file and INQUIRE on it.

INQUIRE for terminals

The INQUIRE TERMINAL command for CICS terminal resources returns named items of information about a particular terminal. See also "Chapter 4.7. Modifying the terminal control table" on page 237, which describes the terminal control macro instruction interface (DFHTC CTYPE macros).

For a remote terminal, EXEC CICS INQUIRE TERMINAL obtains information from the definition of the terminal as a remote terminal. It does not obtain information from the remote system, and in this way differs from the EXEC CICS ASSIGN and EXEC CICS ADDRESS commands.

```
EXEC CICS INQUIRE
          {TERMINAL(4-character data-value)|
           NETNAME(8-character data-value)}
          [NETNAME(8-character data-area)|
           TERMINAL(4-character data-area)]
          [REMOTESYSTEM(4-character data-area)]
          [MODENAME(8-character data-area)]
          [TRANSACTION(4-character data-area)]
          [TERMPRIORITY(fullword binary data-area)]
          [USERAREA(31-bit pointer)]
          [USERAREALEN(halfword binary data-area)]
          [OPERID(3-character data-area)]
          [USERID(8-character data-area)]
          [DEVICE(cvda)]
          [TERMMODEL(halfword binary data-area)]
          [ACCESSMETHOD(cvda)]
          [CREATESESS(cvda)]
          [ACQSTATUS(cvda)]
          [SERVSTATUS(cvda)]
          [ATISTATUS(cvda)]
          [TTISTATUS(cvda)]
          [PAGESTATUS(cvda)]
          [SCREENHEIGHT(halfword binary data-area)]
          [SCREENWIDTH(halfword binary data-area)]
          [GCHARS(halfword binary data-area)]
          [GCODES(halfword binary data-area)]
```

Options

Fuller details of these options are given under their equivalent names in the DFHTCT section of the *CICS/MVS Resource Definition (Macro)* manual.

ACCESSMETHOD

Possible CVDA values are VTAM, BSAM, BTAM, BGAM, TCAM, TCAMSNA, CONSOLE, or NOTAPPLIC. The access method may be VTAM, or, if associated with a LINE, may be BSAM, BTAM, BGAM, TCAM or TCAMSNA. It may also be CONSOLE.

| **ACQSTATUS (VTAM only)**

Possible CVDA values are ACQUIRED, RELEASED, NOTAPPLIC, or ACQUIRING. They indicate whether CICS is in session with the logical unit represented by the terminal.

ATISTATUS

Possible CVDA values are ATI and NOATI, which indicate whether the terminal is available for use by transactions which are automatically initiated from within CICS or, if the terminal is an ISC session, by transactions that are using this session as an alternate facility to communicate with another system.

A terminal cannot have both NOATI and NOTTI in its status.

CREATESESS (VTAM only)
CVDA values CREATE, NOCREATE or NOTAPPLIC indicate whether the terminal can be acquired automatically by ATI transactions.

DEVICE
Identifies the terminal or session type as recorded in the TCTTE. It is a CVDA field, and can be tested using the built-in function provided.

GCHARS
The graphic character set global identifier (GCSGID) which is a registered number from 1 through 65534 representing the set of graphic characters which can be input or output at the terminal.

GCODES
The code page global identifier (CGPID) which is a registered number between 1 and 65534 representing the EBCDIC code page defining the code points for the characters which can be input or output at the terminal.

The term "coded graphic character set identifier" is commonly used when referring to both the above registered numbers taken as a pair.

MODENAME (LU6.2 only)
The name of a group of parallel sessions (of which that named in the TERMINAL field is one), which have similar characteristics.

NETNAME
The name of the terminal or session, as known to VTAM. This will be blanks if the access method is not VTAM.

Either TERMINAL or NETNAME must be the first option used as a sender field. The value given is used as the search argument.

Whichever is specified first, the other may be optionally specified as the second option. This will always be a receiver field.

For parallel sessions the NETNAME will not be unique; in these cases the TERMINAL field will contain the name of the first session found.

OPERID
The operator identification code to be used when signing on to this terminal.

PAGESTATUS
Possible CVDA values PAGEABLE and AUTOPAGEABLE indicate whether pages after the first in a series will be written to the terminal either upon request from the operator, or automatically.

REMOTESYSTEM
If the subject of the inquiry is a session, REMOTESYSTEM returns the name of the associated remote system. If the subject of the inquiry is a remote terminal, REMOTESYSTEM returns the name of the link to the terminal-owning system. If the subject of the inquiry is a local terminal, REMOTESYSTEM is returned as blanks.

SCREENHEIGHT
The height of the current 3270 screen.

The value returned depends on the terminal's mode (DEFAULT or ALTERNATE) at the time the INQUIRE command is processed.

SCREENWIDTH
The width of the current 3270 screen.

The value returned depends on the terminal's mode (DEFAULT or ALTERNATE) at the time the INQUIRE command is processed.

SERVSTATUS
CVDA value INSERVICE or OUTSERVICE indicates whether the terminal is available for use. For LU6.2, INSERVICE means it is valid to attempt to acquire the terminal.

TERMINAL
The terminal name as defined in the TCT. This includes all terminals and sessions defined in the TCT, but not LDCs, surrogate TCTTEs, mode groups, or system entries.

TERMMODEL
Gives the terminal model number.

TERMPRIORITY
Priority of the terminal relative to other terminals, in a range from 0 to 255.

TRANSACTION
The name of the transaction currently executing with the named terminal as its principal facility.

TTISTATUS
CVDA value TTI or NOTTI indicates whether the terminal is available for use by transactions that are initiated from this terminal.

A terminal cannot have both NOATI and NOTTI in its status.

USERAREA
Points to the TCTUA, which contains the process control information (PCI) for this terminal.

USERAREALEN
Contains the length of the user area.

USERID
A security code identifying the person signed on, defined in the signon table.

Exceptional condition

The following condition can occur:

TERMIDERR
The named terminal or netname could not be located.

Browse operations for terminals

The following commands enable the application program to browse through the terminal definitions in the TCT:

```
EXEC CICS INQUIRE TERMINAL START

EXEC CICS INQUIRE TERMINAL(4-character data-area) NEXT
                       (other options as in
                        above option list)

EXEC CICS INQUIRE TERMINAL END
```

Browsing is not terminated by user syncpoints.

INQUIRE TERMINAL START command sets an internal CICS pointer to the first terminal in the TCT. Subsequent invocations of INQUIRE TERMINAL NEXT commands can be used to scan all known terminals.

The following condition can occur:

ILLOGIC A browse is already in progress.

INQUIRE TERMINAL NEXT command returns the next terminal or session name in the area addressed by the TERMINAL keyword.

On input all specified operands are ignored; they are all set on output.

The order of browsing is controlled by a CICS-internal cursor. In particular it does **not** guarantee collating sequence. However, continued browsing does guarantee to return all terminal and session entries in the TCT.

Only one TERMINAL browse operation is allowed at any one time in a given task.

Note that no facility is provided for selective browsing; that is, for returning entries only if they meet certain conditions.

The following conditions can occur:

END The INQUIRE command has located all of the terminals in the system.

ILLOGIC An INQUIRE TERMINAL START has not been issued.

The **INQUIRE TERMINAL END** command ends the browsing operation and frees any resources held.

The following condition can occur:

ILLOGIC An INQUIRE START has not been issued.

SET for terminals

The SET command for CICS terminal resources modifies specified attributes of a particular terminal. See also "Chapter 4.7. Modifying the terminal control table" on page 237, which describes the terminal control macro instruction interface (DFHTC CTYPE macros).

The SET TERMINAL command does not apply to LU6.2 sessions.

```
EXEC CICS SET
     TERMINAL(4-character data-value)
     [SERVSTATUS(cvda)]
     [ACQSTATUS(cvda)]
     [CREATESESS(cvda)]
     [ATISTATUS(cvda)]
     [TTISTATUS(cvda)]
     [PAGESTATUS(cvda)]
     [TERMPRIORITY(fullword binary data-area)]
     [PURGE [FORCE]]
```

Options

The functions that may be SET are defined as follows:

ACQSTATUS

Possible CVDA values are ACQUIRED, RELEASED, and COLDACQ. Setting a terminal to RELEASED will cause the session to be terminated, immediately if PURGE is also specified, or otherwise when the current active transaction has finished.

COLDACQ is a special form of ACQUIRED to be used where no resynchronization is required.

PURGE

PURGE FORCE causes any transaction running with the terminal to be terminated abnormally. With PURGE termination will only occur if system and/or data integrity can be maintained.

SERVSTATUS

Possible CVDA values are INSERVICE and OUTSERVICE. Setting a terminal to OUTSERVICE means that the terminal can no longer be used by transactions. If PURGE or FORCEPURGE is not specified, the transaction will be allowed to terminate normally, but no further transactions will be allowed to use the terminal. However, if you are using EDF on the specified terminal, EDF will stop immediately, because it is a sequence of separate transactions, while the transaction being tested will complete.

For VTAM, setting a terminal to OUTSERVICE will also cause it to be released and the operator signed off, either immediately or when the current transaction has terminated. It is therefore invalid to try to set the terminal associated with the executing transaction to OUTSERVICE, unless it is a printer.

For the keywords below, see the definitions under "INQUIRE for terminals" on page 439:

- ATISTATUS
- CREATESESS
- PAGESTATUS
- TERMPRIORITY
- TTISTATUS.

Exceptional conditions

The following conditions can occur:

ERROR Internal CICS error in terminal control.

TERMIDERR
The SET command could not locate the named terminal.

INVREQ An attempt was made to set an invalid CVDA value, or the terminal is not in a valid state to allow the requested changes.

The RESP2 field contains further information:

Value	Meaning
1	ACQSTATUS specified for a connection that is not LU6.2
2	ACQUIRED specified for a terminal in OUTSERVICE state (LU6.2 conflict only)
3	NOTPENDING specified for a connection that is not LU6.2
4	ATISTATUS problem: invalid CVDA value
5	ATISTATUS problem: would result in NOATI or NOTTI
6	CREATESESS problem: LU6.2, or IRC, or not VTAM
7	CREATESESS problem: invalid CVDA value
8	PAGESTATUS problem: LU6.2 or IRC
9	PAGESTATUS problem: invalid CVDA value
10	SERVSTATUS problem: LU6.2, or IRC, or OUTSERVICE
11	SERVSTATUS problem: trying to put this terminal OUTSERVICE (if it is not a printer)
12	SERVSTATUS problem: trying to put this CSNL terminal OUTSERVICE
13	SERVSTATUS problem: invalid CVDA value
14	TERMPRIORITY problem: LU6.2
15	TERMPRIORITY problem: invalid value
16	TTISTATUS problem: LU6.2 or IRC
17	TTISTATUS problem: trying to make this terminal NOTTI
18	TTISTATUS problem: invalid CVDA value
19	PURGE or FORCE problem: LU6.2 or not VTAM
20	PURGE or FORCE problem: trying to purge or force task on **this** terminal

INQUIRE for system entries (CONNECTIONS)

The INQUIRE command for CICS system entries returns named items of information about connections to a particular system.

```
EXEC CICS INQUIRE
     CONNECTION(4-character data-value)
     [NETNAME(8-character data-area)]
     [ACCESSMETHOD(cvda)]
     [PROTOCOL(cvda)]
     [ACQSTATUS(cvda)|CONNSTATUS(cvda)]
     [SERVSTATUS(cvda)]
     [PENDSTATUS(cvda)]
     [XLNSTATUS(cvda)]
```

Options

Fuller details of these options are given under their equivalent names in the DFHTCT section of the *CICS/MVS Resource Definition (Macro)* manual.

ACCESSMETHOD

Possible CVDA values are VTAM, IRC, INDIRECT, and XM, which indicate that either VTAM, CICS IRC, INDIRECT or cross-memory communication is in use for this connection.

ACQSTATUS (VTAM only)

CVDA values ACQUIRED or RELEASED indicate whether CICS is in session with the logical unit represented by this connection. CONNSTATUS provides more information about the connection. It cannot be used with ACQSTATUS.

CONNECTION (sender)

The four-character system name (as defined in a DFHTCT TYPE=SYSTEM macro) or connection name (as defined using CEDA) for ISC or IRC.

CONNSTATUS (LU6.2 only)

CVDA values ACQUIRED, RELEASED, OBTAINING, FREEING, or AVAILABLE indicate the state of the session, where:

ACQUIRED	The connection is ACQUIRED, which means that: • The partner LU has been contacted, and • Initial CNOS exchange has been done.
RELEASED	The connection is released.
OBTAINING	The connection is being acquired. INQUIRE only.
FREEING	The connection is being released. INQUIRE only.
AVAILABLE	The connection is acquired but there are no bound sessions as they have all been unbound because of limited resource. INQUIRE only.

CONNSTATUS cannot be used with ACQSTATUS.

NETNAME

The name by which the remote system is known to the VTAM network.

PENDSTATUS (LU6.2 only)
CVDA value PENDING or NOTPENDING indicates whether there are pending units of work.

PROTOCOL
CVDA values APPC, LU61, and NOTAPPLIC indicate whether LU6.1, APPC (LU6.2) or something else is the SNA protocol used for this connection.

SERVSTATUS
CVDA value INSERVICE or OUTSERVICE indicates whether the system is available for use. For LU6.2, INSERVICE means the system can be acquired.

XLNSTATUS (LU6.2 only)
CVDA values XOK, XNOTDONE or NOTAPPLIC indicate the status of the exchange lognames (XLN) process, as follows:

XNOTDONE (LU6.2 only)
The exchange lognames (XLN) flow for the LU6.2 connection has not completed successfully; the CSMT log may contain information relating to this state. XNOTDONE means that synclevel(2) conversations are not allowed on the connection (but synclevel(0) and synclevel(1) are still allowed).

XOK (LU6.2 only)
The exchange lognames (XLN) process for the LU6.2 connection has completed successfully.

NOTAPPLIC
One of the following conditions exists:
- The connection is released.
- The connection is MRO, LU6.1, or single session LU6.2.
- The connection does not support synclevel 2 conversations.

Exceptional condition

The following condition can occur:

SYSIDERR The named system entry could not be located.

Browse for system entries (CONNECTION)

The following commands enable the application program to browse through the CONNECTION entries (system entries) in the TCT:

```
EXEC CICS INQUIRE CONNECTION START

EXEC CICS INQUIRE CONNECTION(4-character data-area) NEXT
         (other options as in option list above)

EXEC CICS INQUIRE CONNECTION END
```

Browsing is not terminated by syncpoints.

INQUIRE CONNECTION START command results in an internal pointer being set to the first CONNECTION entry in the TCT.

The following condition may occur:

ILLOGIC A browse is already in progress.

INQUIRE CONNECTION NEXT command returns the next CONNECTION entry name in the area addressed by the CONNECTION keyword. On input all specified operands are ignored; they are all set on output.

Browsing does not necessarily follow collating sequence. However, continued browsing does return all CONNECTION entries in the TCT. Only one CONNECTION browse operation is allowed at any one time in a given task.

Note that there is **no** facility for selective browsing; that is, for returning entries only if they meet certain conditions.

The following conditions can occur:

END The INQUIRE command has located all of the SYSTEM/CONNECTION entries in the TCT.

ILLOGIC An INQUIRE CONNECTION START has not been issued.

INQUIRE CONNECTION END command ends the browsing operation and free any resources held.

The following condition can occur:

ILLOGIC An INQUIRE START has not been issued.

SET for system entries (CONNECTION)

The SET function allows certain attributes of the SYSTEM/CONNECTION entry to be modified. Note that control returns to the issuing program when the required operation has been started, not necessarily when it has completed.

CICS uses a temporary storage queue with the default data identifier (dataid) prefix DF when acquiring and releasing LU6.2 sessions. If temporary storage dataids with the prefix DF are defined as recoverable in your installation, you must follow the SET CONNECTION command by a SYNCPOINT command to end the logical unit of work and allow the SET CONNECTION command to complete.

```
EXEC CICS SET
     CONNECTION(4-character data-value)
     [ACQSTATUS(cvda)|CONNSTATUS(cvda)]¹
     [SERVSTATUS(cvda)]
     [NOTPENDING]
     [PURGE [FORCE]]
```

¹Only the ACQUIRED and RELEASED CVDA values are available with CONNSTATUS

Options

See the option list under "INQUIRE for system entries (CONNECTIONS)" on page 446 for the meanings of these options. For PURGE, see "SET for terminals" on page 444.

Exceptional conditions

The following conditions can occur:

INVREQ An attempt was made to set an invalid CVDA value, or the connection is not in a valid state for the requested changes.

The RESP2 field contains further information:

Value	Meaning
1	ACQSTATUS or CONNSTATUS was specified for a connection that is neither LU6.2 nor VTAM
2	ACQUIRED conflicts with OUTSERVICE (LU6.2 conflict only)
3	ACQSTATUS or CONNSTATUS has an invalid CVDA value
4	SERVSTATUS has an invalid CVDA value
5	NOTPENDING was specified for a connection that is not LU6.2
6	PURGE was specified for a connection that is not VTAM

SYSIDERR The named SYSTEM/CONNECTION entry could not be located.

INQUIRE for modenames

The INQUIRE command for a CICS modename returns named items of information about a mode group defined for a particular system connection.

```
EXEC CICS INQUIRE
     MODENAME(8-character data-value)
     CONNECTION(4-character data-area)
     [MAXIMUM(halfword binary data-area)]
     [AVAILABLE(halfword binary data-area)]
     [ACTIVE(halfword binary data-area)]
```

Note that both MODENAME and CONNECTION are required in this command.

Options

Fuller details of these options are given under their equivalent names in the DFHTCT section of the *CICS/MVS Resource Definition (Macro)* manual.

ACTIVE
The number of sessions within the group which are actually bound (that is, in use). This may vary between 0 and the number available.

AVAILABLE
The number of sessions within the group that may be concurrently allocated for use. During a CICS run you may dynamically change this value, between 0 and the maximum.

CONNECTION (sender)
The four-character remote system name in which the specified modename is defined.

MAXIMUM
The maximum number of concurrent sessions supported within the mode group.

MODENAME (sender)
The eight-character name of a group of sessions defined for a particular system connection. These names are unique within the connection, but need not be unique within the whole of CICS.

Exceptional condition

The following condition can occur:

SYSIDERR The named connection could not be located, or the modename within the connection could not be located.

Browse for modenames

Browse operations are allowed for MODENAME information.

The following commands enable the application program to browse through the MODENAME entries in the TCT:

```
EXEC CICS INQUIRE MODENAME START

EXEC CICS INQUIRE MODENAME(8-character data-area) NEXT
                       CONNECTION(4-character data-value)
                       (other options as in
                        option list above)

EXEC CICS INQUIRE MODENAME END
```

Browsing is not terminated by user syncpoints.

INQUIRE MODENAME START command will result in internal pointers being set to the first MODENAME entry in the first CONNECTION defined in the TCT.

The following condition may occur:

ILLOGIC A browse is already in progress.

INQUIRE MODENAME NEXT command requires the CONNECTION to be specified as a sender field. It may be null.

The INQUIRE NEXT command returns:

The next modename within this connection, if the CONNECTION is the same as on the previous INQUIRE NEXT. Or, if there are no more, it returns the first modename in the next connection which has modenames.

The first modename within this connection, if the CONNECTION is different from the previous one.

On input all the other operands are ignored; they are all returned on output.

The order of browsing does **not** guarantee collating sequence. However continued browsing does guarantee to return all MODENAME entries within all CONNECTIONs in the TCT. Only one MODENAME browse operation is allowed at any one time in a given task.

The following conditions can occur:

END	The INQUIRE command has located all of the modename entries in the TCT.
ILLOGIC	An INQUIRE MODENAME START has not been issued.
SYSIDERR	The named CONNECTION could not be located within the TCT.

INQUIRE MODENAME END command ends the browsing operation and frees any resources held.

The following condition can occur:

ILLOGIC An INQUIRE START has not been issued.

SET for modenames

The SET function allows the number of **available** sessions associated with this modename to be changed.

If the number is decreased, the excess number of sessions will be unbound.

```
EXEC CICS SET
     MODENAME(8 character data-value)
     CONNECTION(4-character data-value)
     [AVAILABLE(halfword binary data-area)]
     [ACQUIRED]
```

Options

For AVAILABLE and CONNECTION, see also options under "INQUIRE for modenames" on page 449.

ACQUIRED
If the number of available sessions is increased, this option causes the extra sessions to be bound.

CONNECTION (sender)
CONNECTION must be specified to qualify the modename.

Exceptional conditions

The following conditions can occur:

SYSIDERR The RESP2 field contains this information:

Value	Meaning
1	The named connection could not be located.
2	The modename within the connection could not be located.

INVREQ The RESP2 field contains this information:

Value	Meaning
3	The modename 'SNASVCMG' was specified.This is a reserved name and may not be set.
4	The specified AVAILABLE value is not in the range 0 to MAXIMUM
5	The specified connection is not in session.

INQUIRE for system attributes

INQUIRE support is provided to give access to system information.

```
EXEC CICS INQUIRE SYSTEM
          [RELEASE(4-character data-area)]
          [OPSYS(1-character data-area)]
          [OPREL(halfword binary data-area)]
          [MAXTASKS(fullword binary data-area)]
          [AMAXTASKS(fullword binary data-area)]
          [AKP(fullword binary data-area)]
          [CUSHION(fullword binary data-area)]
          [TIME(fullword binary data-area)]
          [RUNAWAY(fullword binary data-area)]
          [STALL(fullword binary data-area)]
```

Options

AKP

The activity keypoint trigger value. It is the number of system logging operations between the taking of keypoints. This value cannot be changed if AKPFREQ=0 was specified at CICS initialization.

AMAXTASKS

The maximum number of concurrent active tasks. The journal and terminal control tasks are outside the scope of this limit, and should not be allowed for in assigning the value.

CUSHION

The level of unassigned storage which defines the "storage cushion". On setting, the given value is rounded up to the next page size (2048 or 4096).

MAXTASKS
The maximum number of concurrent tasks allowed to run in the CICS system, including both active and suspended tasks.

OPSYS
Identifies the type of operating system currently running. It returns the 1-character value 'X', meaning 'XA'.

OPREL
Indicates the release number of the operating system currently running. It is a halfword binary integer equal to 10 times the formal release number.

RELEASE
Identifies the level of the CICS system. The value returned is a four-character string in the form *vvrm*, where *vv* is the version number, *r* is the release number, and *m* is the modification number. For example, the value "0210" represents CICS/MVS 2.1.0, "0211" represents CICS/MVS 2.1.1, and "0212" represents CICS/MVS 2.1.2.

RUNAWAY
The maximum length of time for which a task can have control before it is assumed to be in a runaway condition (logical loop). After this interval a task is abnormally terminated. This is task-time only, not real time.

STALL
The value of the stall time-interval. If no active CICS tasks have been able to proceed during a period equal to the stall time-interval, the system is considered to have stalled. Purgeable tasks will then be abended to reduce the load on the system.

TIME
The maximum interval in milliseconds after which CICS will relinquish control to the operating system if no transactions are ready to resume execution. This is known as the region exit time interval.

SET for system attributes

```
EXEC CICS SET SYSTEM
              [MAXTASKS(fullword binary data-value)|
              AMAXTASKS(fullword binary data-value)|
              AKP(fullword binary data-value)|
              CUSHION(fullword binary data-value)|
              TIME(fullword binary data-value)|
              RUNAWAY(fullword binary data-value)|
              STALL(fullword binary data-value)]
```

Options

For meanings of these options, see option list under "INQUIRE for system attributes" on page 452.

Only one value may be changed in one SET command, to avoid the complicated and order-dependent checking and error-handling that would otherwise be necessary.

When you set RUNAWAY, STALL or TIME, the value is rounded down to the nearest 10. The values that you can set for each option must be within the limits given under the INVREQ exceptional condition.

The recording of each system value takes place immediately, but its effect is only seen when the relevant resource manager accesses the particular value.

Exceptional condition

After a SET SYSTEM command, the following condition can occur:

INVREQ An attempt was made to do one of the following:

- SET MAXTASKS to < 2 or < AMAXTASKS or > 999 or <= the largest CMAXTASK value.
- SET AMAXTASKS to < 1 or > 999 or > MAXTASKS.
- SET AKP to < 200 or > 65,535 or to anything if AKPFREQ = 0.
- SET CUSHION < 0 or > 524,288.
- SET TIME < 100 or > 327,670.
- SET RUNAWAY < 500 or > 2,700,000.
- SET STALL < TIME or > 327,670.

INQUIRE for programs

This command enables you to INQUIRE about the attributes of programs, maps, and partition sets, using the following command:

```
EXEC CICS INQUIRE
          PROGRAM(8-character data-value)
          [LANGUAGE(cvda)]
          [PROGTYPE(cvda)]
          [STATUS(cvda)]
          [LENGTH(fullword-binary data-area)]
          [RESCOUNT(fullword-binary data-area)]
          [USECOUNT(fullword-binary data-area)]
```

Options

LANGUAGE
: Possible CVDA values of COBOL, PLI or PL1, or ASSEMBLER indicate the language in which the program was written.

LENGTH
: The length of the program in bytes. A value of 0 is returned if the program has not been used in the current CICS session.

PROGRAM (sender)
: The name of the program, map, or partition set as defined in the PPT.

PROGTYPE
: Possible CVDA values are PROGRAM, MAP, or PARTITIONSET. Maps, partition sets, and programs are listed in the PPT. This field indicates the type of entry in the PPT.

RESCOUNT
: The number of times the program is currently being used.

STATUS
: CVDA values of ENABLED and DISABLED indicate whether the program is available for use.

USECOUNT
: The number of times the program has been used in the current CICS session.

Exceptional conditions

The following conditions can occur:

PGMIDERR	The named program could not be located.
NOTAUTH	The invoker is not authorized to access the program.

Browse for programs

You can use browse operations to access information about programs, maps, and partition sets. You can only examine information about programs that you are authorized to access. The commands are as follows:

```
EXEC CICS INQUIRE PROGRAM START

EXEC CICS INQUIRE PROGRAM(8-character data-area) NEXT
                         (other options as in
                          option list above)

EXEC CICS INQUIRE PROGRAM END
```

Browsing is not terminated by syncpoints.

INQUIRE PROGRAM START command will result in an internal pointer being set to the first PROGRAM entry in the PPT.

ILLOGIC A browse is already in progress.

INQUIRE PROGRAM NEXT command returns information about the next PPT entry in the specified area.

On input all specified operands are ignored; they are all set on output.

The order of browsing is strictly undefined and in particular does **not** guarantee collating sequence. However, continued browsing does guarantee to return all authorized entries in the PPT. Only one PROGRAM browse operation is allowed at any one time in a given task.

Note that **no** facility is provided for selective browsing; that is, for returning entries only if they meet certain conditions.

The following conditions can occur:

END The INQUIRE command has located all of the authorized entries in the PPT.

ILLOGIC An INQUIRE PROGRAM START has not been issued.

INQUIRE PROGRAM END command ends the browsing operation and frees any resources held.

The following condition can occur:

ILLOGIC An INQUIRE START has not been issued.

SET for programs

The SET command enables you to alter some of the attributes of program entries in the PPT. The command is as follows:

```
EXEC CICS SET
        PROGRAM(8-character data-value)
        [STATUS(cvda)]
        [NEWCOPY]
```

Options

NEWCOPY

marks the program as nonresident. NEWCOPY can therefore be used to obtain a newly link-edited version of the program, or to restore a program that has been overwritten. If NEWCOPY is specified for a program that is currently executing, the INVREQ condition will be raised.

STATUS

Possible CVDA values are ENABLED and DISABLED. Programs beginning with "DFH" cannot be disabled.

Exceptional conditions

The following conditions can occur:

PGMIDERR The named program could not be located.

NOTAUTH The invoker is not authorized to access the program.

INVREQ The RESP2 field contains this information:

Value	Meaning
1	Trying to disable a DFH... program
2	STATUS problem: invalid CVDA problem
3	NEWCOPY problem: program is executing

ERROR This is a NEWCOPY problem. The load failed either because it was being loaded already, or because no new copy was available.

INQUIRE for transactions

This command enables you to INQUIRE on transaction attributes.

```
EXEC CICS INQUIRE
          TRANSACTION(4-character data-value)
          [REMOTESYSTEM(4-character data-area)]
          [PROGRAM(8-character data-area)]
          [STATUS(cvda)]
          [PRIORITY(fullword binary data-area)]
```

Options

PRIORITY
The priority of this transaction relative to other transactions, in a range from 0 through 255.

PROGRAM
The name of the program to be executed first when this transaction is started.

REMOTESYSTEM
The name of the remote system on which the transaction is defined. If LOCAL, this value will be blanks.

STATUS
A CVDA value of ENABLED or DISABLED indicates whether the transaction is available for use.

TRANSACTION (sender)
The transaction name defined in the PCT.

Exceptional conditions

The following conditions can occur:

TRANSIDERR The named transaction could not be located.

NOTAUTH The named transaction is not authorized to the user.

Browse for transactions

This browse facility only returns information about transactions if the invoking program is authorized to access that transaction. You use the following commands:

```
EXEC CICS INQUIRE TRANSACTION START

EXEC CICS INQUIRE TRANSACTION(4-character data-area) NEXT
                       (other options as in
                        option list above)

EXEC CICS INQUIRE TRANSACTION END
```

Browsing is not terminated by syncpoints.

INQUIRE TRANSACTION START command will result in an internal pointer being set to the first TRANSACTION entry in the PCT.

The following condition can occur:

ILLOGIC A browse is already in progress.

INQUIRE TRANSACTION NEXT command accesses the information about transactions listed in the PCT.

On input all specified operands are ignored; they are all set on output.

The order of browsing is strictly undefined and, in particular, does **not** guarantee collating sequence. However, continued browsing does guarantee to return all authorized entries in the PCT. Only one TRANSACTION browse operation is allowed at any one time in a given task.

Note that **no** facility is provided for selective browsing; that is, for returning entries only if they meet certain conditions.

The following conditions can occur:

END The INQUIRE command has returned all the authorized entries in the PCT.

ILLOGIC An INQUIRE TRANSACTION START has not been issued.

INQUIRE TRANSACTION END command ends the browsing operation and frees any resources held.

The following condition can occur:

ILLOGIC
An INQUIRE START has not been issued.

SET for transactions

This command gives the ability to SET enabled and priority status of transactions, as defined in the PCT.

```
EXEC CICS SET
        TRANSACTION(4-character data-value)
        [STATUS(cvda)]
        [PRIORITY(fullword binary data-area)]
        [PURGEABILITY(cvda)]
```

Options

See options under "INQUIRE for transactions" on page 457 for PRIORITY option.

PURGEABILITY
Possible CVDA values of PURGEABLE and NOTPURGEABLE override the SPURGE value held in the PCT. This option causes the transaction to be marked "purgeable" or "notpurgeable" for reference in system stall conditions.

STATUS
A CVDA value of ENABLED or DISABLED indicates whether the transaction is available for use. Transactions beginning with "C" cannot be disabled.

Exceptional conditions

The following conditions can occur:

TRANSIDERR The named transaction could not be located.

INVREQ The RESP2 field contains this information:

Value	Meaning
1	PRIORITY problem: value out of range
2	PURGEABILITY problem: invalid CVDA value
3	STATUS problem: invalid CVDA value
4	Transactions beginning with "C" cannot be disabled.

NOTAUTH The named transaction is not authorized to the user.

CICS-VALUES used in INQUIRE and SET commands

The CICS-values in CVDAs returned by the INQUIRE command or sent by the SET command are listed here under two headings:

- CICS-values for conditions, attributes, and languages
- CICS-values for device types.

See page 424 for a more detailed explanation of CICS-value data-areas and the DFHVALUE function.

CICS-values for conditions, attributes, and languages

Figure 40 on page 461 lists the CICS-values for conditions, attributes, and languages.

```
NOTAPPLIC     EQU   1         IGNORE        EQU   1
BDAM          EQU   2         VSAM          EQU   3
REMOTE        EQU   4         ESDS          EQU   5
KSDS          EQU   6         RRDS          EQU   7
KEYED         EQU   8         NOTKEYED      EQU   9
BASE          EQU  10         PATH          EQU  11
FIXED         EQU  12         VARIABLE      EQU  13
UNDEFINED     EQU  14         BLOCKED       EQU  16
UNBLOCKED     EQU  17         OPEN          EQU  18
CLOSED        EQU  19         OPENING       EQU  20
CLOSING       EQU  21         CLOSEREQUEST  EQU  22
ENABLED       EQU  23         DISABLED      EQU  24
DISABLING     EQU  25         OLD           EQU  26
SHARE         EQU  27         NEW           EQU  28
RECOVERABLE   EQU  29         NOTRECOVABLE  EQU  30
EMPTYREQ      EQU  31         NOEMPTYREQ    EQU  32
UNENABLED     EQU  33         UNENABLING    EQU  34
READABLE      EQU  35         NOTREADABLE   EQU  36
UPDATABLE     EQU  37         NOTUPDATABLE  EQU  38
BROWSABLE     EQU  39         NOTBROWSABLE  EQU  40
ADDABLE       EQU  41         NOTADDABLE    EQU  42
DELETABLE     EQU  43         NOTDELETABLE  EQU  44
HEX           EQU  45         DEC           EQU  46
BLK           EQU  47         EXCTL         EQU  48
NOEXCTL       EQU  49         VTAM          EQU  60
BSAM          EQU  61         BTAM          EQU  62
BGAM          EQU  63         TCAM          EQU  64
TCAMSNA       EQU  65         CONSOLE       EQU  66
CREATE        EQU  67         NOCREATE      EQU  68
ACQUIRED      EQU  69         RELEASED      EQU  70
ACQUIRING     EQU  71         COLDACQ       EQU  72
INSERVICE     EQU  73         OUTSERVICE    EQU  74
ATI           EQU  75         NOATI         EQU  76
TTI           EQU  77         NOTTI         EQU  78
PAGEABLE      EQU  79         AUTOPAGEABLE  EQU  80
FREEING       EQU  94         AVAILABLE     EQU  95
OBTAINING     EQU  96         IRC           EQU 121
INDIRECT      EQU 122         XM            EQU 123
APPC          EQU 124         LU61          EQU 125
PENDING       EQU 126         NOTPENDING    EQU 127
XOK           EQU 143         XNOTDONE      EQU 144
ASSEMBLER     EQU 150         COBOL         EQU 151
PLI           EQU 152         PL1           EQU 152
PROGRAM       EQU 154         MAP           EQU 155
PARTITIONSET  EQU 156         PURGEABLE     EQU 160
NOTPURGEABLE  EQU 161
```

Figure 40. CICS-values for conditions, attributes, and languages

CICS-values used for device types

Figure 41 lists the CICS-values used for device types.

T7770	EQU 1	SYSTEM7	EQU 2
TCONSOLE	EQU 8	SEQDISK	EQU 18
MAGTAPE	EQU 20	CDRDLPRT	EQU 24
HARDCOPY	EQU 32	TWX33/35	EQU 33
TELETYPE	EQU 34	T1050	EQU 36
T2740	EQU 40	T2741COR	EQU 42
T2741BCD	EQU 43	VIDEOTERM	EQU 64
T2260L	EQU 65	T2260R	EQU 72
T1053	EQU 74	T2265	EQU 76
TTCAM	EQU 80	BISYNCH	EQU 128
T2770	EQU 130	T2780	EQU 132
T3780	EQU 133	T2980	EQU 134
T3735	EQU 136	T3740	EQU 137
T3600BI	EQU 138	T3277R	EQU 145
T3275R	EQU 146	T3284R	EQU 147
T3286R	EQU 148	T3277L	EQU 153
T3284L	EQU 155	T3286L	EQU 156
BIPROG	EQU 160	SYSTEM3	EQU 161
SYS370	EQU 164	SYS7BSCA	EQU 166
SDLC	EQU 176	T3601	EQU 177
T3614	EQU 178	T3790	EQU 180
T3790UP	EQU 181	T3790SCSP	EQU 182
T3650PIPE	EQU 184	T3653HOST	EQU 185
T3650ATT	EQU 186	T3650USER	EQU 187
CONTNLU	EQU 189	INTACTLU	EQU 190
BATCHLU	EQU 191	LUTYPE6	EQU 192
LUTYPE4	EQU 193	RESSYS	EQU 208
ISCMMCONV	EQU 209	LUCMODGRP	EQU 210
LUCSESS	EQU 211		

Figure 41. CICS-values used for device types

EIBRCODEs of the INQUIRE and SET commands

The following are the return codes of the INQUIRE and SET commands.

EIBFN (Byte 0)	Byte (of EIBRCODE)	EIBRCODE Value	Condition
4C	3	0C	FILENOTFOUND
4C	3	10	INVREQ
4C	3	11	IOERR
4C	3	15	ILLOGIC
4C	3	46	NOTAUTH
4C	3	53	END
4E	3	01	ERROR
4E	3	10	INVREQ
4E	3	15	ILLOGIC
4E	3	1B	PGMIDERR
4E	3	46	NOTAUTH
4E	3	53	END
50	3	10	INVREQ
50	3	15	ILLOGIC
50	3	1C	TRANSIDERR
50	3	46	NOTAUTH
50	3	53	END
52	3	01	ERROR
52	3	0B	TERMIDERR
52	3	10	INVREQ
52	3	15	ILLOGIC
52	3	53	END
54	3	10	INVREQ
58	3	10	INVREQ
58	3	15	ILLOGIC
58	3	35	SYSIDERR
58	3	53	END
5A	3	10	INVREQ
5A	3	15	ILLOGIC
5A	3	35	SYSIDERR
5A	3	53	END

Chapter 5.10. CICS interface to JES

General introduction

The CICS interface to JES (the Job Entry Subsystem component of MVS) provides specialist system programmer EXEC commands for accessing the system spool files maintained by JES2 and JES3. You can support the requirements of other products to exchange files with other systems connected through a JES RSCS (remote spooling communications subsystem) network. The term JES is used to refer to both JES2 and JES3.

The CICS interface to JES enables you to:

1. Retrieve data for a specific user from the local JES spool. See Figure 42 on page 467.
2. Create a file and write records directly to the local JES spool. See Figure 43 on page 468.
3. Send a JES spool file to a specific remote destination. See Figure 44 on page 468.

There are certain internal limits in JES2 and JES3 that you should consider when you are designing applications to use this interface. Some of these internal limits may depend on which release of JES you are using. You should therefore read the following in conjunction with the appropriate JES documentation.

JES2

- Number of SYSOUT data sets

  There is an upper limit to the number of data sets that can be created by a single job. If this limit is exceeded during a CICS run, subsequent SPOOLOPEN OUTPUT requests will return the ALLOCERR exceptional condition.

- Output queue and job queue sizes

  The number of Job Output Elements and Job Queue Elements may need to be increased to accommodate the additional output processing. Timely processing of the datasets created using this interface will minimize this requirement.

- Spool space

  Although the spool space for a data set created using this interface is reused after it has been processed, some control information is retained for the life of the job. You may have to increase the spool file allocation to allow for this.

JES3

- Job queue size

  The number of Job Queue Elements may need to be increased to accommodate the additional output processing. Also, more JSAM buffers may be required. Timely processing of the data sets created using this interface will minimize this requirement.

- Spool space

  Although the spool space for a data set created using this interface is reused after it has been processed, some control information is retained for the life of the job. You may have to increase the spool file allocation to allow for this.

For both JES2 and JES3, some performance degradation may be experienced if a backlog of CICS-created data sets is allowed to accumulate. You should ensure that procedures exist to detect and remedy such situations.

Input

A remote application must route any files intended for a CICS transaction to a specific user name at the system where CICS resides. See Figure 42 on page 467 for an example of a CP command used by a VM system to do this. The figure also shows the EXEC CICS SPOOL commands you use to retrieve the data.

The CICS transaction issues the SPOOLOPEN command, specifying that writer name on the USERID parameter and optionally the class of output within the writer name. The normal response is:

1. No input for this external writer.
2. The single-thread is busy (see below).
3. The file is allocated to you for retrieval, and is identified by the "token" returned by CICS. The token must be included on every SPOOL command for retrieving the data set.

(See "Exceptional conditions and RESP2 values" on page 476 for a full list of responses.)

In cases (1) and (2) the transaction should retry the SPOOLOPEN after a suitable interval, by restarting itself.

In case (3) the transaction should then retrieve the file with SPOOLREAD commands, and proceed to SPOOLCLOSE as rapidly as possible to release the path for other users. This is especially important for **input** from JES because the input path is **single-thread**. When there is more than one transaction using the interface, their files may be differentiated by using different writer names or different classes within a single writer name. Furthermore, you should ensure that the transactions either terminate or WAIT for a short period between SPOOLCLOSE and a subsequent SPOOLOPEN. If you do not do this, one transaction may prevent others from using the interface.

JES exits

Both JES2 and JES3 provide a way of screening incoming files. For JES2, the TSO/E Interactive Data Transmission Facility Screening and Notification exit is used. The JES3 equivalent is the Validate Incoming Netdata File exit.

You should review any use your installation makes of these exits to ensure that files that are to be read using the CICS interface to JES are correctly processed.

Output

The transaction program issues SPOOLOPEN to cause allocation of an output data set, specifying a remote NODE and USERID. SPOOLOPEN returns a unique token to the transaction, which must be used in all subsequent SPOOLWRITE and SPOOLCLOSE commands to identify the file being written to. Finally, the transaction issues SPOOLCLOSE to close and deallocate the report, and permit its immediate printing or onward routing by the system spooler. The normal response received from the SPOOLOPEN OUTPUT command is:

> The file is allocated to you and identified by the token returned by CICS. Data may now be written to it.

If the node is a remote MVS system, then the data set will be queued on the JES spool against the external writername. The id of this writername was specified on the SPOOLOPEN OUTPUT USERID parameter. If the node is a remote VM system, then the data is queued in the VM RDR queue for the id that was specified on the same USERID parameter.

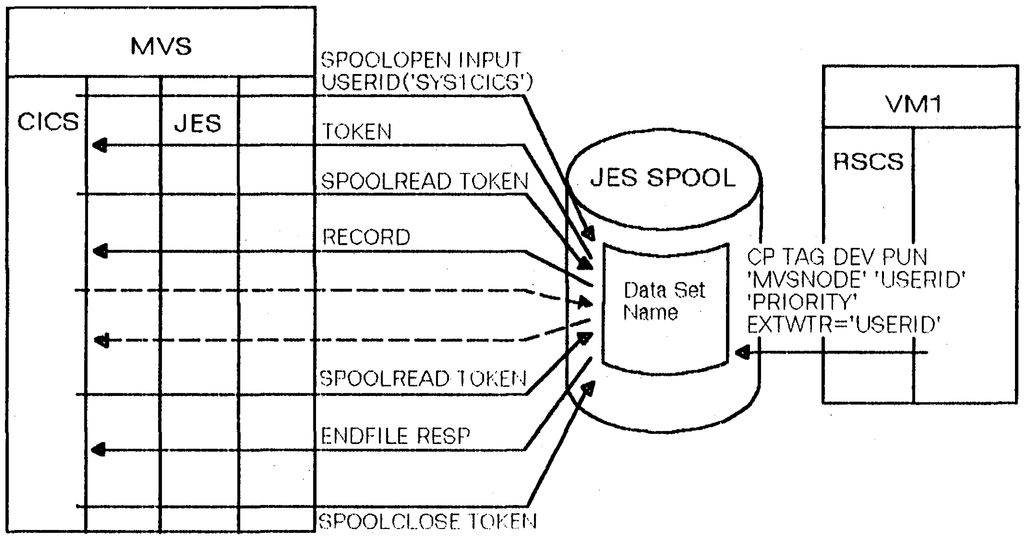


Figure 42. Retrieve data from the JES spool

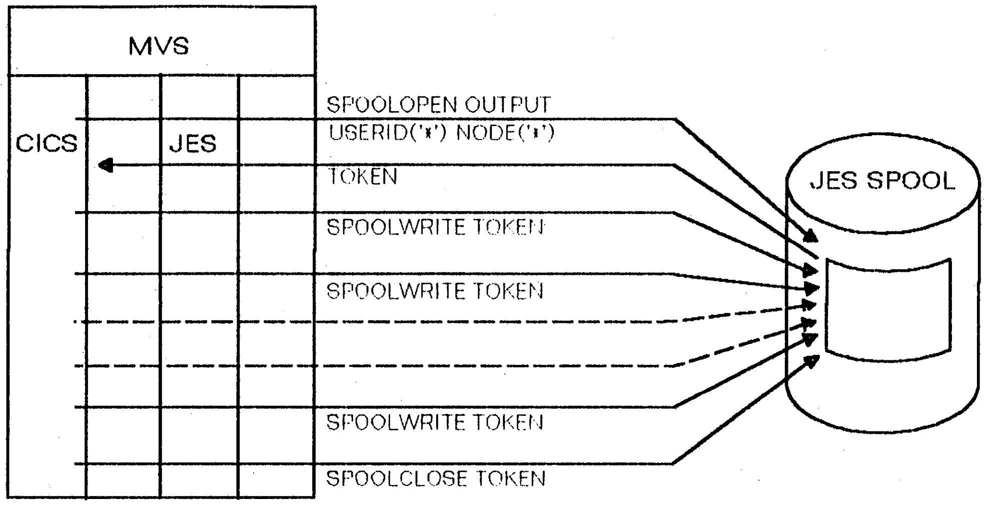


Figure 43. Create a file and write directly to the JES spool. See also Figure 44.

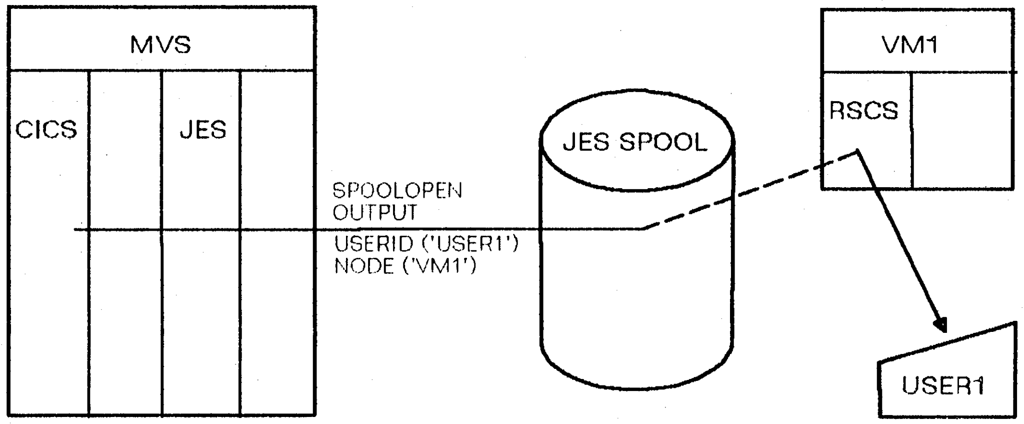


Figure 44. Send the written file to a remote destination

Typical use

The following example is a demonstration of the EXEC CICS SPOOL commands:

1. The spool file is opened. If there is no data, or the single-thread input path is busy, the program sends an error message and restarts in five seconds.
2. If there is data and the input file is available, the program opens the output file. Then it reads and writes the logical records back to the output file.
3. At the comment PROCESS DATA, a report could be generated. For clarity, only comments have been inserted here.
4. The input and output files are closed.

```
DFHEISTG DSECT
CICSAMPL CSECT
OPENIN   EQU   *
* OBTAIN AND CLEAR GETMAIN AREA
         EXEC CICS GETMAIN SET(PTR) LENGTH(500) INITIMG(BLANK)
         USING RENTAREA,PTR
* OPEN INPUT SPOOLFILE
         EXEC CICS SPOOLOPEN INPUT USERID(XWTRNAME)                    *
               TOKEN(INTOKEN)                                          *
               RESP(RESPFLD) RESP2(RESP2FLD)
         CLC   RESPFLD,DFHRESP(NORMAL)     OPEN OK?
         BE    OPENOUT             YES
         CLC   RESPFLD,DFHRESP(SPOLBUSY)   NO, IS SPOOL IN USE?
         BE    RESTART             YES, TRY AGAIN LATER
         CLC   RESPFLD,DFHRESP(NOTFND)     NO, IS ANY DATA ON SPOOL?
         BE    RESTART             NO,  TRY AGAIN LATER
OPENERR  EQU   *  ERROR TERMINATE TRANSACTION AND RETURN MESSAGE
         MVC   MSGAREA(L'ERRMSG1),ERRMSG1
         B     SAMPLEOJ
RESTART  EQU   *  SPOOL BUSY OR NO DATA. TRY AGAIN LATER.
         EXEC CICS START TRANSID('LADM') INTERVAL(000005)              *
               TERMID('BLA3') NOCHECK PROTECT
         MVC   MSGAREA(L'OKMSG02),OKMSG02
         B     SAMPLEOJ
OPENOUT  EQU   * EVERYTHING OK. DATA ON INPUT FILE. OPEN OUTPUT.       *
         EXEC CICS SPOOLOPEN OUTPUT NODE(VM1) USERID(USER1)            *
               TOKEN(OUTTOKEN) NOCC                                    *
               RESP(RESPFLD) RESP2(RESP2FLD)
         CLC   RESPFLD,DFHRESP(NORMAL)     OPEN OK?
         BE    CONTINU0            YES
         MVC   MSGAREA('ERRMSG2),ERRMSG2     NO! GIVE ERROR MSG
         B     SAMPLEOJ
CONTINU0 EQU   *     READ INPUT DATA AND PROCESS
         EXEC CICS SPOOLREAD INTO(INREC)                               *
               TOKEN(INTOKEN)                                          *
               TOFLENGTH(L80) MAXFLENGTH(L132)                         *
               RESP(RESPFLD) RESP2(RESP2FLD)
         CLC   RESPFLD,DFHRESP(NORMAL)      READ OK?
         BE    CONTINU1             YES
         CLC   RESPFLD,DFHRESP(ENDFILE)     ANY MORE DATA?
         BE    ENDATA               NO
         MVC   MSGAREA(L'ERRMSG3),ERRMSG3    MOVE IN ERROR MSG
         B     SAMPLEOJ
CONTINU1 EQU   *                   YES, PROCESS DATA
* PROCESS DATA. CREATE REPORT AND PREPARE TO OUTPUT REPORT
* TO SPOOLER OUTPUT FILE.
         MVI   CTLCHAR,C' '
         MVC   OUTDATA,CTLCHAR
         MVC   OUTDATA,INREC
         EXEC CICS SPOOLWRITE FROM(OUTREC)                             *
               TOKEN(OUTTOKEN)                                         *
               FLENGTH(L80)                                            *
               RESP(RESPFLD) RESP2(RESP2FLD)
         CLC   RESPFLD,DFHRESP(NORMAL)      WRITE OK?
         BE    CONTINU2             YES
         MVC   MSGAREA('ERRMSG4),ERRMSG4    NO MOVE ERROR MSG
         B     SAMPLEOJ
CONTINU2 EQU   *
         B     CONTINU0      READ NEXT INPUT RECORD FROM SPOOL FILE
ENDATA   EQU   *
* CREATE TOTAL SALE PAGE AND OUTPUT TO SPOOLER FILE.
* CLOSE INPUT AND OUTPUT SPOOL FILES.
CLOSEIN  EQU   *
         EXEC CICS SPOOLCLOSE TOKEN(INTOKEN)                          *
               RESP(RESPFLD) RESP2(RESP2FLD) DELETE
         CLC   RESPFLD,DFHRESP(NORMAL)    CLOSE INPUT OK?
         BE    CLOSEOUT             YES
         MVC   MSGAREA(L'ERRMSG5A),ERRMSG5A    NO MOVE ERROR MSG
         MVC   MSGAREA+23(L'ERRMSG5B),ERRMSG5B
         MVC   MSGAREA+36(L'ERRMSG5D),ERRMSG5D
CLOSEOUT EQU   *
         EXEC CICS SPOOLCLOSE TOKEN(OUTTOKEN)                         *
               RESP(RESPFLD) RESP2(RESP2FLD) KEEP
         CLC   RESPFLD,DFHRESP(NORMAL)    CLOSE OUTPUT OK?
         BE    SAMPLEOJ             YES
         MVC   MSGAREA(L'ERRMSG5A),ERRMSG5A    NO MOVE ERROR MSG
         MVC   MSGAREA+28(L'ERRMSG5C),ERRMSG5C
         MVC   MSGAREA+36(L'ERRMSG5D),ERRMSG5D
SAMPLEOJ EQU   *
         MVC   MSGAREA+50(L'OKMSG01),OKMSG01   MOVE EOJ MESSAGE
         EXEC CICS SEND TEXT FROM(MSGAREA) LENGTH(80)
         EXEC CICS RETURN
R1       EQU   1             REGISTER 1
PTR      EQU   5             REGISTER 5
OKMSG01  DC    CL20' TRAN LADM EOJ.     '
OKMSG02  DC    CL45' TRAN LADM RESTARTED, NO DATA OR SPOOL BUSY. '
ERRMSG1  DC    CL45' TRAN LADM TERMINATED. SPOOLOPEN INPUT ERROR.'
ERRMSG2  DC    CL46' TRAN LADM TERMINATED. SPOOLOPEN OUTPUT ERROR.'
ERRMSG3  DC    CL39' TRAN LADM TERMINATED. SPOOLREAD ERROR.'
ERRMSG4  DC    CL40' TRAN LADM TERMINATED. SPOOLWRITE ERROR.'
ERRMSG5A DC    CL23' TRAN LADM, SPOOLCLOSE '
ERRMSG5B DC    CL5'INPUT'
ERRMSG5C DC    CL7' OUTPUT'
ERRMSG5D DC    CL6'ERROR.'
L80      DC    F'80'
L132     DC    F'132'
BLANK    DC    X'00'
XWTRNAME DC    CL8'CICSAMPL'
USER1    DC    CL8'USER1   '
VM1      DC    CL8'VM1     '
RENTAREA DSECT
INTOKEN  DS    CL8
OUTTOKEN DS    CL8
RESPFLD  DS    CL4
RESP2FLD DS    CL4
INREC    DS    CL80
OUTREC   DS    0CL133
CTLCHAR  DS    CL1
OUTDATA  DS    CL132
MSGAREA  DC    CL80'
         END
```

EXEC CICS commands

The EXEC CICS commands listed in this section give access to the system spooler. They are intended for system programmer use only.

To use the CICS interface to JES you must code DFHSIT SPOOL=YES.

You must specify RESP or NOHANDLE on these commands. RESP bears a one-to-one correspondence with HANDLE CONDITION. If you do not code RESP your program will abend. You can also code the RESP2 option. NOHANDLE, RESP and RESP2 are not shown in the syntax boxes or in the option lists. See page 424 for information about the use of RESP and RESP2. At the end of that chapter is a list of RESP values and the RESP2 values that are CICS-specific to the CICS-JES interface.

Transactions that process SYSOUT data sets larger than 1000 records, either for INPUT or OUTPUT, are likely to have a performance impact on the rest of CICS. When you cannot avoid such a transaction, you should carefully evaluate general system performance. You should introduce a pacing mechanism if the effects on the rest of CICS are unacceptable.

See "Options" on page 473 for descriptions of all the options that apply to these EXEC CICS commands.

All access to a JES spool file must be completed within one logical unit of work. Issuing an EXEC CICS SYNCPOINT command will implicitly SPOOLCLOSE any open report.

OPEN for input

```
EXEC CICS SPOOLOPEN
        INPUT
        USERID(ext_writer_name)
        TOKEN(token)
        [CLASS(class)]
```

This command opens a spool report for input from the system spooler to CICS.

It prepares to get (read) an existing spool data set directly using external_writer_name (USERID) and specified class.

Another task may have allocated a spool file for input. In this case, the user should retry after a suitable time interval.

When this command has been successfully executed, users should read the report and proceed to CLOSE as soon as possible, in order to permit other users to use the JES single thread. If SPOOLCLOSE is not issued before transaction end or SYNCPOINT, CICS performs an implicit SPOOLCLOSE KEEP, and writes a message to CSMT to alert the system programmer to the possible unnecessary retention of resources. You should not SPOOLOPEN a data set using this command until you are prepared to process it completely.

This command, if successful, will return a token, which is used later to identify the report in SPOOLREAD and SPOOLCLOSE commands.

OPEN for output

```
EXEC CICS SPOOLOPEN
          OUTPUT
          NODE(node_id)
          USERID(userid)
          [CLASS(class)]
          TOKEN(token)
          [NOCC|ASA|MCC]
          [PRINT|PUNCH]
          [OUTDESCR(address)]
```

This command opens a spool report for output from CICS to the system spooler and defines its characteristics.

It results in a dynamic allocation of the output file using the node_id to specify the remote destination and the user_id to specify the remote user. As this is a multithread output request, requestors of this service could interleave. This SPOOLOPEN enables users to acquire the token for a report that it expects to create (write). This token is used to identify the report in later SPOOLWRITE and SPOOLCLOSE commands.

When printing on a local device, use the NOCC|ASA|MCC parameters to control output formatting. If you do not specify a format, the default value of NOCC is used. NODE('*') and USERID('*') may be used to write the data set directly to the local spool file. For example:

```
EXEC CICS SPOOLOPEN OUTPUT NODE('*') USERID('*') CLASS('A') ASA
```

If you do not issue SPOOLCLOSE before the end of the transaction, CICS performs an implicit SPOOLCLOSE DELETE and writes a message to CSMT to alert you to the possible unnecessary retention of resources.

Note: If you retrieve a formatted data set, the system spooler may have changed the data set format. For example, the system spooler may have converted an MCC format data set to ASA format during data set creation. This does not affect the the final printed output.

READ record

```
EXEC CICS SPOOLREAD
          TOKEN(token)
          INTO(ioarea)
          [TOFLENGTH(length)]
          [MAXFLENGTH(length)]
```

This command obtains the next record from the system spooler.

WRITE record

```
EXEC CICS SPOOLWRITE
        TOKEN(token)
        FROM(io_area)
        [FLENGTH(length)]
        [LINE|PAGE]
```

This command writes data to a spool report.

CLOSE a report

```
EXEC CICS SPOOLCLOSE
        TOKEN(token)
        [KEEP|DELETE]
```

This command closes a spool report and optionally changes its retention characteristics. If more than one transaction is trying to read reports from JES, then SPOOLCLOSE should **not** be immediately followed by SPOOLOPEN. It should be followed by a WAIT, so that other transactions may use the interface.

Options

All options below, other than pure keywords, refer to data areas. Receivers, senders, and optional or alternative fields are identified.

ASA
This specifies that the report about to be created will have each record prefixed with an ASA carriage-control character, and that this character must be used by the operating system to control formatting when the report is printed.

CLASS(class)
is the class designation. This assigns a class to the output data set, and may be used as a selection parameter for the input report. CLASS is a one-character sender field. CLASS is optional. If it is omitted on input then the first report for the specified writer will be obtained, regardless of its class. If it is omitted on output then class A is assumed.

FLENGTH(length)
Specifies the length of data transferred as a fullword binary number. This is set by the user on output. It is optional, and if it is omitted CICS will use the length of the data area.

FROM(ioarea)
Specifies the data area from which variable length data will be transferred. The data itself is not altered in any way by CICS. FROM is a sender field.

INPUT
Indicates that the report is to be read by CICS.

INTO(ioarea)
Specifies the data area into which variable-length data will be transferred. It is a receiver field.

KEEP|DELETE
Specifies the disposition code.

If an INPUT report is closed with disposition KEEP, it will be read again when SPOOLOPEN INPUT is next issued. Closing it with DELETE will result in the next report being read on the subsequent OPEN INPUT. KEEP is the default if the user fails to close the report. However if the SPOOLCLOSE command is issued without a KEEP or DELETE option, then DELETE is assumed for input reports.

For an OUTPUT report, the KEEP option is ignored. The DELETE option purges a report. When neither option is specified, the report goes to its destination node.

LINE|PAGE
PAGE means that you are writing page-mode data, and will cause the CPDS indicator to be set in the ACB for the write request. If you do not want the CPDS indicator to be set, you should allow this value to default to LINE.

MAXFLENGTH(length)
Specifies the maximum length of data to be read as a fullword binary number. This is an optional value, and if it is omitted CICS will use the length of the data area. If you do provide a MAXFLENGTH value, CICS acquires a buffer of the size that you specify. The MAXFLENGTH value must not be more than 32KB minus 8 bytes.

MCC
This specifies that the report about to be created will have each record prefixed with an IBM machine command code carriage-control character, and that this character must be used by the operating system to control formatting when the report is printed.

NOCC
This specifies that the report about to be created will have no internal formatting controls. When the report is printed, the operating system will prefix each record with a carriage-control character that will cause page skipping according to the default operating-system lines-per-page value.

NODE(node_id) and USERID(userid)
Node_id is the eight-character ID of a destination node which the system spooler will use to route the file. It is a sender field.

Userid is the identifier of the eventual writer program or user who will process the report. The report will carry this identifier and it will be used to select the report at its destination. It is a sender field. For an INPUT report, the userid must begin with the same four characters as the CICS APPLID, so that CICS can check that the user is not attempting to access data sets not intended for his CICS system.

Code NODE('*') and USERID('*') to specify the local spool file and to enable the OUTDESCR operand to override the NODE and USERID operands. If the NODE and USERID operands specify explicit identifiers, the OUTDESCR operand cannot override them.

OUTDESCR(address) (MVS/SP*-JES2 Version 3 only)

The address points to a field that contains the address of a string consisting of parameters to the OUTPUT statement of MVS JCL. The user must set up the pointer, the address field, and the string. The format of the string is:

Offset	Length	Contents
0	4	Length (n) of following text string
4	n	OUTPUT statement parameters

The parameters use the same keywords and values as the OUTPUT statement but the syntax varies slightly. The following is the format of the OUTDESCR parameter string:

```
keyword1(value1) [keyword2(value2)] [keyword3(value3,value4)] ...
```

This corresponds to the following OUTPUT statement parameter string:

```
keyword1=value1 [keyword2=value2] [keyword3=(value3,value4)] ...
```

For details of valid keywords and values, see *MVS/ESA Job Control Language Reference*, GC28-1829-0.

The OUTDESCR operand:

- Can override the NODE and USERID operands only if they are specified as NODE('*') and USERID('*') respectively.
- Cannot override the CLASS operand, even if it is omitted and defaults to class A.

Use this operand to set additional attributes for the spool data set. For example, to associate a forms ID with the spoolfile for a local JES2-managed printer, you could code:

```
EXEC CICS SPOOLOPEN NODE('*') USERID('*') CLASS('A') OUTDESCR(PARMS)
```

PARMS is a BLL cell (OS/VS COBOL) or a pointer (VS COBOL II, PL/I). For Assembler, PARMS is a register containing the address of an address constant. If the required OUTDESCR parameter is 'FORMS(WIDE)', PARMS points to an address field (of similar type) that points to a 15-byte area containing the value 11 in bytes 1-4, and the string 'FORMS(WIDE)' in bytes 5-15.

VS COBOL II users who wish to use the linkage section for this purpose must issue EXEC CICS GETMAIN for storage to hold the keyword structure, and then set addressability to it using the VS COBOL II SET command.

OUTPUT

Indicates that the report is to be written by CICS.

* IBM Trademark. For a list of trademarks see page iii.

PRINT
Is included for compatibility with the spool support provided with CICS/DOS/VS, and to allow large records to be written to the spool. PRINT is the default setting for the SPOOLOPEN OUTPUT command, and it has no effect on the SPOOLOPEN INPUT command.

PUNCH
Is included for compatibility with the spool support provided with CICS/DOS/VS. You must specify this parameter if the CLASS parameter for the output data set implies punch, and the data set is destined for a VM/RSCS node. This ensures that the record length indicator is set to 80, which is a requirement of VM/RSCS for punch files. The PUNCH parameter has no effect on the SPOOLOPEN INPUT command.

TOFLENGTH(length)
Specifies the length of data transferred as a fullword binary number. This is set by CICS on input. It is optional, and if it is omitted the user will not be notified of the actual length of the data received.

TOKEN(token)
This is the CICS-allocated token used to identify a report. It is an eight-byte value aligned on a fullword boundary. It is a receiver on SPOOLOPEN and a sender on all other commands.

Exceptional conditions and RESP2 values

Here is a list of exceptional conditions that can occur in response to the spool commands listed above. Under some of the exceptional conditions are listed RESP2 values that are specific to the CICS commands for the interface to JES. (You may encounter RESP2 values other than those documented here for use by the CICS interface. These are provided by JES-MVS or VSAM for other purposes.)

ALLOCERR
MVS dynamic allocation has rejected a request to allocate an input data set.

RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in *OS/VS2 MVS Programming Library: Job Management*.

ENDFILE
Indicates that all data for the current spool file being SPOOLREAD has been retrieved. You should proceed to issue a SPOOLCLOSE command as soon as possible, to release the lock on the JES single thread, and to terminate current SYSOUT data set processing.

ILLOGIC
You have specified an invalid parameter.

RESP2 value	Meaning of RESP2 value
3	Invalid CLASS parameter specified

INVREQ
Invalid request.

RESP2 value	Meaning of RESP2 value
4	Unsupported language
8	Unsupported function
12	Read attempt after end of file
16	USERID missing
20	NODE missing
24	INTO missing
28	FROM missing
32	KEEP|DELETE OUTPUT specified
36	INPUT|OUTPUT missing
40	CICS SSI already enabled
44	Bad OUTDESCR string
48	OUTDESCR specified but function not available (wrong level of MVS or JES)
52	OUTDESCR specified, but bad pointer found in keyword or in OUTDESCR list

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM support center.

LENGERR

For a SPOOLREAD command, this conditions results from one of the following (if the buffer space is too small, as much data as possible will be read):

- You requested insufficient buffer space to SPOOLREAD your record
- You requested more than the maximum allowable (32KB—8 bytes).

For a SPOOLWRITE command, this conditions results from one of the following:

- The value specified in the FLENGTH parameter command was zero.
- The value specified in the FLENGTH parameter command was negative
- The value specified in the FLENGTH parameter command was greater than the maximum record length value that was used when the spool was opened for output.

RESP2 indicates the amount of data truncated, the difference between the FLENGTH value and the maximum length value at SPOOLOPEN for output time, or shows zero if the MAXFLENGTH field is greater than 32KB minus 8 bytes or if the FLENGTH field contains a zero or a negative value.

NODEIDERR

JES cannot identify the USER/NODE combination specified on SPOOLOPEN OUTPUT.

RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code, (S99INFO), and the second two are the error reason code, (S99ERROR), as defined in *OS/VS2 MVS Programming Library: Job Management*.

NOSPOOL
The JES interface is not available.

RESP2 value	Meaning of RESP2 value
4	No subsystem present
8	Interface being disabled; CICS is quiescing
12	Interface has been stopped

NOSTG
An OS GETMAIN has failed within the JES interface subtask (DFHPSPSS).

RESP2 gives the GETMAIN register 15 return code.

NOTAUTH
Indicates that an application has issued a SPOOLOPEN INPUT command with an unauthorized USERID. For the USERID to be authorized, its first four characters must match the first four characters of the current CICS system id.

NOTFND
Indicates that either data, or specific internal control block signals, could not be located.

RESP2 value	Meaning of RESP2 value
4	No data sets could be located for retrieval for the specified external writer name
1024	Input or output function has been corrupted, and SPOOLCLOSE could not complete

NOTOPEN
Indicates an input/output error. Failure of attempt to SPOOLREAD from, or SPOOLWRITE to, or SPOOLCLOSE an unopened (SPOOLOPEN) data set.

RESP2 value	Meaning of RESP2 value
8	Data set has not been opened, or a task that did not issue the SPOOLOPEN for a spool data set has attempted to access it
12	Attempt to read an output file
16	Attempt to write an input file
1024	Subtask OPEN macro failure

OPENERR
An internal error occurred during SPOOLOPEN processing that has forced the request to fail.

RESP2 value	Meaning of RESP2 value
4	A VSAM SHOWCB macro failed to return the lengths of the VSAM control blocks used to access the JES spool file

OUTDESCRERR

The MVS macro, OUTADD or OUTDEL (invoked as a result of the OUTDESCR specification) failed.

RESP2 gives the reason code from the OUTADD or OUTDEL macro.

SPOLBUSY

Non-availability of the JES/input single thread within the JES interface.

RESP2 value	Meaning of RESP2 value
4	Interface already in use by another task
8	Interface already in use by current task

SPOLERR

The MVS subsystem interface macro (IEFSSREQ) has failed. No input data set name was selected.

RESP2 gives the IEFSSREQ response code.

STRELERR

An OS FREEMAIN has failed within the JES interface subtask (DFHPSPSS).

RESP2 gives the FREEMAIN register 15 return code.

EIBRCODEs of the system spooler commands

The following are the return codes of the system spooler commands. The EIBFN (Byte 0) value is the first byte of the associated EIBFN code.

```
┌EIBFN (Byte 0)
│
│   ┌Byte (of EIBRCODE)
│   │
│   │  ┌EIBRCODE Value
│   │  │
│   │  │   ┌Condition
│   │  │   │
56  3  0D  NOTFND
56  3  10  INVREQ
56  3  13  NOTOPEN
56  3  14  ENDFILE
56  3  16  LENGERR
56  3  2A  NOSTG
56  3  46  NOTAUTH
56  3  50  NOSPOOL
56  3  55  ALLOCERR
56  3  56  STRELERR
56  3  57  OPENERR
56  3  58  SPOLBUSY
56  3  59  SPOLERR
56  3  5A  NODEIDERR
```

End of General-Use Programming Interface

Chapter 5.11. Finding programs that use CICS macros

If you are a large installation and you want to convert your CICS applications to command-level, you must first convert all your CICS macro-level programs. To help in this task, CICS/MVS 2.1.2 provides an DFHMSCAN program that scans a load module library and identifies programs that use CICS macros.

You should investigate any program that DFHMSCAN identifies as having CICS macro-level requests. The program's language is identified from the last macro call analyzed. Therefore, if PL/I or COBOL programs contain assembler routines with CICS macro-level calls (for example, if a COBOL or PL/I program has library modules link-edited to it, and these contain CICS macro-level calls), the program as a whole may be flagged as assembler.

This is the JCL you need:

```
//SCANJOB  JOB ACCOUNTING INFO,CLASS=A
//SCAN     EXEC PGM=DFHMSCAN,PARM='pppppppp'
//STEPLIB  DD DSN=CICS212.LOADLIB,DISP=SHR
//INPUT    DD DSN=xxxxxxx.LOADLIB,DISP=SHR
//OUTPUT   DD SYSOUT=A
//SUMMARY  DD SYSOUT=A
//
```

`xxxxxxx.LOADLIB` is the load module library to be scanned.
pppppppp, the PARM in the EXEC statement has two possible values to specify the processing and report required:

PARM='$SUMMARY'
DFHMSCAN scans every module in the load library, and produces an overall report. This is the default action if PARM is not coded. See "Summary report."

PARM='NAME1,NAME2,...'
DFHMSCAN scans the named modules and produces a detailed report for each one. See "Detailed report."

Summary report

If you specify `PARM='$SUMMARY'`, DFHMSCAN:

1. Scans each module in the load library for BALR 14,14 and BALR 14,15 instructions.
2. Analyzes the instructions preceding identified BALR instructions, to see if they match sequences used by CICS macro requests or EXEC CICS commands.
3. Prints, for each module in the scanned library, its name, size, language (if determined), the number of CICS macro-level statements, the number of CICS command-level statements, and the number of unrecognized BALR instructions.
4. Prints the total number of modules in the library and the number of macro-level programs of each type (assembler, COBOL and PL/I).

Detailed report

If you specify PARM='NAME1,NAME2,...', DFHMSCAN scans the named modules and for each module:

1. Prints a line for each BALR found, giving its offset from the start of the module, some of the code that precedes it and what it appears to be.
2. For macro-level statements, attempts to identify the macro type.

Restrictions

- DFHMSCAN does not scan CICS modules and tables in the load library, or any modules that are loaded above the 16 MB line. (DFHMSCAN is linkedited with AMODE(24)).
- DFHMSCAN does not separately identify CHECK macros.
- DFHMSCAN cannot identify certain forms of the DFHBIF macro that do not produce a BALR, or that produce code indistinguishable from that generated by EXEC CICS commands.
- For COBOL and PL/I, because the code depends so much on the compiler, DFHMSCAN cannot find all EXEC CICS commands. The main purpose of DFHMSCAN is to find macro-level programs.
- DFHMSCAN can only find code patterns that are similar to those generated by CICS macros. A module can contain such code without having a CICS macro in its source.

Chapter 5.12. CEMT programming interface

Compatibility note

The format of CEMT commands can change between releases, and as a result of maintenance (APARs). Upwards compatibility is not guaranteed for programs using the interface described in this chapter. If the function you require is available through EXEC CICS INQUIRE and SET commands, it is recommended that you use those commands.

This chapter describes the programmable interface to the master terminal transaction, CEMT. The functions provided by the master terminal transaction can be invoked from application programs, by a command such as:

```
EXEC CICS LINK PROGRAM('DFHEMTA')
               COMMAREA(CEMTPARM)
```

where DFHEMTA is the name of the entry point in the master terminal program, and CEMTPARM is a user-defined name of a parameter list consisting of five 24-bit addresses (each contained in a fullword) as follows:

1. Address of a field containing the master terminal command in source form.
2. Address of a halfword binary field specifying the length of the command. The maximum length of the input command is 1022 bytes.
3. Address of a one-byte indicator field defined as follows:

   X'80' — display output at terminal instead of returning it to caller.
4. Address of a field in which output is to be placed by DFHEMTA.
5. Address of a halfword binary field specifying the maximum length of output that the application can handle.

If the indicator in address 3 is X'80', output is displayed at the terminal. In this case, you can enter any number of CEMT commands at the terminal in the usual way. Control is returned to the program when you press PF3.

If the indicator is X'00' (that is, output is not to be displayed at the terminal) DFHEMTA returns control to your application program immediately after processing the master terminal command specified in the first address. At the same time, DFHEMTA returns the output as one or two concatenated, structured fields. Each field has the format:

- Binary halfword containing inclusive length of field.
- Binary halfword containing:
  - For the translation stage — number of messages produced.
  - For the execution stage — number of resources.

- Binary halfword containing:
  - For the translation stage — highest message-severity, 0, 4, and 8 will continue to execution, 12 will not continue to execution.
  - For the execution stage — code for last response that was not normal. See EIERM001 and so on in DFHEIMDS for the meanings of each code.
- Variable-length data containing:
  - For the translation stage — diagnostic messages if there are any.
  - For the execution stage — one line of data for each resource. Each line begins with a new line (NL) character, but otherwise consists of blanks and uppercase alphanumerics.

The format of this data is not guaranteed from release to release, but is the same as displayed by CEMT. Note that the response of 'Normal/Error' displayed by CEMT is not included in this data. (Analysis of this data should not normally be necessary if commands referring to more than one resource are avoided.)

The output from a single request comprises one field for the translation stage and one or none for the execution stage. If the total output is longer than the maximum length specified by the user, it will be truncated.

Note: An attempt to start CEMT from an application program by either an EXEC CICS START command or a DFHIC TYPE=INITIATE macro will fail. This is because CEMT's first action is to request input from its associated terminal, whereas an automatically initiated transaction must first send data to the terminal.

An attempt to start CEMT under CECI by an EXEC CICS START command will fail for similar reasons. A PERFORM SHUTDOWN should not be issued using the programmable interface because a terminal is required for successful completion.

Part 6. Files and data sets

This part of the book contains reference information on certain operations that you may perform on files and data sets.

"Chapter 6.1. The explicit open/close function" on page 487 provides information on the DFHOC macro instruction, which you can use to open and close files and data sets during the execution of CICS.

"Chapter 6.2. Dynamic allocation sample program" on page 493 describes the terminal operation procedures for using the sample program, and discusses the keyword values and spellings.

"Chapter 6.3. Loading and accessing files that use phonetic codes for keys" on page 499 describes the function that allows misspelled names to be used as keys to access data sets.

Note: The INQUIRE/SET commands for files are included in "Chapter 5.9. Examining and modifying resource attributes," starting on page 430.

Chapter 6.1. The explicit open/close function

Before looking at the rest of this chapter, read the following notes. CICS provides other commands that perform the same function as DFHOC, as well as an automatic open at first reference.

Notes:

1. The DFHOC macro instruction, described below, is intended for use by system programmers for system control. It should not be used by application programmers to open or close files or data sets.
2. The DFHOC macro is available only to macro-level programs. At command level, you can use the EXEC CICS SET FILE command to open and close those files managed by file control. The SET FILE command has all the advantages of the command-level programming interface, and it allows you to enable and disable your files. See "SET for files" on page 435 in "Chapter 5.9. Examining and modifying resource attributes."
3. You can also use the the CEMT SET FILE command to perform the same functions as the DFHOC OPEN|CLOSE macro. See the *CICS/MVS CICS-Supplied Transactions* manual.
4. A closed enabled file (VSAM or BDAM) is opened automatically at first reference, by both macro-and command-level programs. The automatic open should be the normal way of opening files.
5. Before a file is opened by any of the above methods, it may need to be dynamically allocated if it was not allocated at CICS startup. The dynamic allocation happens automatically if the data set name and disposition information have been set in the file control table at the time of the open. You can set this information as part of the FCT assembly, by CEMT SET FILE or by EXEC CICS SET FILE.

This chapter contains reference information on the DFHOC TYPE=OPEN, CLOSE, and SWITCH macro instructions of the dynamic open/close function.

The CICS dynamic open/close function allows you to open or close files or data sets dynamically during the execution of a CICS macro-level application program. The types of file or data set that may be opened or closed using this function are those that are managed by file control (files), dump management (dump data sets), and transient data management (extrapartition data sets).

The open/close macro instruction (DFHOC) is used to request any of the following services:

- Open or close files
- Open, close, or switch dump data sets
- Open or close transient data extrapartition data sets.

Opening data sets and files — DFHOC TYPE=OPEN

You can open one or more data sets or files by issuing the DFHOC TYPE=OPEN macro instruction.

```
DFHOC   TYPE=OPEN
        ,DATASET={TRANSDATA|DATABASE|DUMP}
        [,CHECK=symbolic-address]
        [,DSETID=(name[,(xx)],...)]
        [,LISTADR={YES|(register)|
                   (symbolic-register)}]
        [,SYMBADR=symbolic-address]
```

TYPE=OPEN
Specifies the open function. If the resource being opened is a file managed by CICS file control, its initial state and final state is as shown in the table below:

Initial state	Final state
closed, enabled	open, enabled
closed, disabled	open, disabled
closed, unenabled	open, enabled

DATASET={TRANSDATA|DATABASE|DUMP}
Specifies the type of resource (file or data set) to be opened.

TRANSDATA
Indicates a transient data extrapartition data set.

DATABASE
Indicates a file managed by file control.

DUMP
Indicates a dump data set.

CHECK=symbolic-address
Specifies the symbolic address of a user-written routine to which control is passed if any error is detected during the OPEN operation. The user-written routine is given control whenever TCAOCTR in the TCA contains a nonzero return code. It is your responsibility to examine the return code in the TCA and, if necessary, examine the individual error codes in the list that was built either by you or as a result of the expansion of the DFHOC macro instruction. The error code appears in the first byte of the third word of each entry in the parameter list.

Upon return from the dynamic open/close program, TCAOCTR may contain one of the following hexadecimal codes:

00	— No error
FF	— Invalid request

or, if TCAOCTR contains neither of these codes, it will contain one or more of the following hexadecimal codes:

80	Open error
40	Close error
20	No space available for OPEN
10	Invalid control block name.

While performing the requested service on the list of files, the individual error bytes in the list entry are filled either with a hexadecimal 00 or with the appropriate error code each time an error is encountered. If more than one error is encountered while processing the parameter list, TCAOCTR reflects all the errors and perhaps a bit configuration different from those shown above. For example, there are six files to be opened; if four are successfully opened, one has an invalid control block identification, and the other one has an open error, the TCAOCTR field contains hex 90.

When there is insufficient storage available to open any files, TCAOCTR contains hex 20, and all the entries contain a fullword (four bytes) of zeros in the third word.

DSETID=name

Specifies the file names or destination identifications to be used in constructing a parameter list. If a suffix is specified, it must be separated from the name or destination identification by a comma, and it must be enclosed in parentheses. This operand is not applicable if DATASET=DUMP is coded, or if LISTADR or SYMBADR is used.

If DATASET=DATABASE is coded, as many as 255 files can be specified with a single use of the DSETID operand. If DATASET=TRANSDATA is coded, up to 255 transient data destination identifications can be specified with a single use of the DSETID operand.

If TYPE=OPEN is coded, and if the destinations are nonresident, "xx," a two-character suffix of the data set control block (DCB) must be provided with each destination identification; if the destination is resident, the "xx" suffix is ignored.

If "xx" consists of more than two characters, it is assumed to be the symbolic address of a list of options and parameters to be moved into the DCB. For the format of this list, see the discussion of the LISTADR operand in this section.

LISTADR={YES|register|symbolic-register}

Specifies the address of the open/close parameter list that you built.

YES

Indicates that the address of the parameter list has been placed in the TCA at TCAOCLA.

register

Indicates the register containing the address of the parameter list.

symbolic register

Indicates the symbolic register name containing the address of the parameter list.

This operand is not applicable if you coded DATASET=DUMP. If the LISTADR and SYMBADR operands are omitted, execution of the DFHOC macro instruction causes the list to be built for you starting with the first byte of the TWA. In this case, it is your responsibility to make sure that the required space is available in the TWA. The space can be calculated using the formula:

space = (n x 12) + 4

where "n" is the decimal number of 12-byte entries in the open/close parameter list, and "4" represents four bytes of hexadecimal Fs to signify the end of the parameter list.

The symbolic storage definition (DFHOCLDS) of a parameter list entry is provided by CICS. The format of the 12-byte entry in the open/close parameter list is:

TRANSDATA

WORD 1	Four-byte destination identification.
WORD 2	Four bytes of the form /"b/"bxx, where /"b/" is two bytes of blanks and xx is a two-byte suffix of the data set control block created by the DCT assembly.
WORD 3	Error byte plus three-byte address of DCT entry (aftercompletion).

DATABASE

WORDS 1 and 2	File name (left justified, padded with blanks).
WORD 3	Error byte plus three-byte address of FCT entry (aftercompletion).

Note: The parameter list must be terminated by hex 'FF'.

You can optionally specify, in WORD 2 of a TRANSDATA entry, the parameter list address pointing to a storage area. This storage area contains information to be placed into a dummy DCB before opening it. If an address is placed in this field, the first byte must be set to hex 'FF'. The symbolic storage definition (DFHOCODS) of this parameter list is provided by CICS. The format of the parameter list is as follows:

Byte 1	Open options byte
Byte 2	BUFNO byte
Byte 3	RECFM byte
Byte 4	ERROPT byte
Bytes 5,6	LRECL
Bytes 7,8	BLKSIZE
Bytes 9-16	DDNAME

The first eight bytes must contain the correct hexadecimal codes for the desired parameters, because the 16 bytes of the open/close parameter list are moved into the DCB.

SYMBADR=symbolic-address
indicates the symbolic address of an open/close parameter list that you built. If the SYMBADR and LISTADR operands are omitted, execution of the DFHOC macro instruction causes the parameter list to be built for you, starting with the first byte of the TWA. For a discussion of the parameter list, see the discussion of the LISTADR operand in this section. This operand is not applicable if DATASET=DUMP is specified.

Closing data sets and files — DFHOC TYPE=CLOSE

You can close one or more files or data sets by issuing the DFHOC TYPE=CLOSE macro instruction. The DATASET, CHECK, LISTADR, and SYMBADR operands have the same significance as in DFHOC TYPE=OPEN. *If a recoverable data set is to be closed, the task should commit any prior changes to the data set. Otherwise file control rejects the request with TCAFCTR=X'20'.*

```
DFHOC   TYPE=CLOSE
        ,DATASET={TRANSDATA|DATABASE|DUMP}
        [,CHECK=symbolic-address]
        [,DSETID=(name,...)]
        [,LISTADR={YES|(register)|
                (symbolic-register)}]
        [,SYMBADR=symbolic-address]
```

TYPE=CLOSE
specifies the close function. If the resource being closed is a file managed by CICS file control, the macro closes the file and unenables it to prevent access by new transactions. The initial state and final state of the file is as shown in the table below:

Initial state	Final state
open, enabled	closed, unenabled
open, disabled	closed, disabled

DSETID=name
specifies the names of the files or data sets to be closed. No suffix is required. As many as 255 names can be specified with a single use of this operand.

Switching dump data sets — DFHOC TYPE=SWITCH

You can switch from the dump data set currently being used to the alternate dump data set by issuing the DFHOC TYPE=SWITCH macro instruction. This macro instruction causes the current dump data set, if open, to be closed, and the alternate dump data set to be opened. A TYPE=CLOSE,DATASET=DUMP macro instruction does not cause a switch, but only closes the current dump data set.

```
DFHOC   TYPE=SWITCH
        ,DATASET=DUMP
```

TYPE=SWITCH
specifies the switch function.

DATASET=DUMP
specifies that the dump data set is to be switched.

Chapter 6.2. Dynamic allocation sample program

The dynamic allocation (DYNALLOC) sample application program makes available to the CICS terminal operator the majority of functions of DYNALLOC (SVC 99). These functions are described fully in the *OS/VS2 MVS System Programming Library: Job Management* manual. Functions that require authorized program facility (APF) authorization are not supported.

The application consists of one command-level assembler language program, DFH99. The source code is provided in CICS212.SAMPLIB.

Using DYNALLOC functions, the terminal operator can dynamically allocate or deallocate any data set that CICS can open and close. With suitable operating discipline and CEMT commands, these can include:

- Extrapartition transient data sets
- Journals
- Dump, trace, and statistics data sets
- DL/I databases. If you have IMS/VS Version 2.2 or IMS/ESA Version 3.1 or later, you can use its dynamic allocation and deallocation support for DL/I.

The dynamic allocation program can also allocate and deallocate data sets that are to be associated with files managed by file control. But you will not normally need this program for files. You can use the dynamic allocation and deallocation facility which is part of CICS. If a file has not been allocated as part of CICS startup, dynamic allocation occurs as a result of, and immediately before, the file is opened, if sufficient information is in the file control table. The information needed is the data set name and disposition of the file. This information is set by the CEMT SET FILE master terminal transaction, described in the *CICS/MVS CICS-Supplied Transactions* manual, or the EXEC CICS INQUIRE and SET commands, which provide additional inquiry and control facilities, described in "Chapter 5.9. Examining and modifying resource attributes," starting on page 430.

In order to use the dynamic allocation sample program effectively, the terminal operator should:

- Have an understanding of MVS job control language, or TSO ALLOCATE and FREE commands.
- Have read the relevant sections of the *OS/VS2 MVS System Programming Library: Job Management* manual and have that manual available for reference while using the sample program, in particular, for looking up error and reason codes returned by DYNALLOC.

The application uses a 3270 display, and adjusts its formatting to suit the screen size. BMS is not required. The program is designed so that the installation may easily modify the functions supported to suit installation standards.

Table entries

Transaction and program definitions for the dynamic allocation sample are provided in the sample utilities group DFH$UTIL on the CSD. These definitions are installed using the CEDA command:

```
CEDA INSTALL GROUP(DFH$UTIL)
```

Alternatively, if you define PCT and PPT entries with the resource definition macros, you can use the table entries provided in the sample tables DFHPCT1$ and DFHPPT1$.

For transaction ADYN:

```
DFHPCT  TYPE=ENTRY,TRANSID=ADYN,PROGRAM=DFH99
```

For program DFH99:

```
DFHPPT  TYPE=ENTRY,PROGRAM=DFH99
```

Note: If you make any changes to the sample program, you must run the DFH99BLD procedure before using the ADYN transaction.

Terminal operation

When transaction ADYN is entered at a terminal, the operator is presented with a formatted display. The top part of the display is for entering commands, the bottom part for receiving messages from the program.

The operator types a command in TSO-like syntax, for example,

```
verb {keyword[(value...)]}...
```

and presses the ENTER key. The program checks the command for correct syntax, builds a DYNALLOC parameter list, and if no serious errors are detected, issues a DYNALLOC SVC. Messages are then displayed to diagnose syntax errors, give the DYNALLOC return codes, and show any values returned by DYNALLOC information retrieval features. The command remains on the display, and the editing features of the terminal may be used to correct it for reentry, or to enter a different command. (If there are too many messages to fit into the message area of the screen, messages that cannot be displayed are queued, and the messages already on the screen are displayed with a brighter intensity to indicate that there are more messages to come. The operator can correct those errors that are being displayed, and reenter the command for further checking, when the queued messages, if any, will be regenerated.

The program is terminated by entering a null command, which consists of pressing the ERASE INPUT key, followed by the ENTER key. PA keys 1 and 2 are ignored by the program. If you press the CLEAR key, you redisplay the last command entered. Pressing a program function key is equivalent to pressing ENTER.

Help feature

The program includes a limited "help" feature, driven by the program's keyword table.

In response to "?", the verb keywords are displayed. In response to "verb?", all the operand keywords of that verb are displayed. For "verb operand(?)" a short description of the value expected for that operand is displayed. When a command containing "?" is entered, no DYNALLOC SVC is issued. "?" is only recognized in the positions specified above; the rest of the command is ignored.

Values

Values are classified as follows:

Keyword value
Keyword values must be specified for some keywords. For example, the STATUS keyword may have a keyword value of SHR, NEW, MOD or OLD (which can be abbreviated).

String of key letters
The value can be a string of letters in any order. The program does not check that the combination of letters provided is meaningful. For example, for the RECFM keyword, the value can be a string of letters from A B D F G M R S T U and V.

Returned values
No value should be provided by the terminal operator, because this keyword requests a value to be returned by the DYNALLOC information retrieval features. The further description refers to the kind of value that will be returned. This is usually in the form in which the operator would enter it, although in a few cases the value is specified as a hexadecimal string.

Not allowed
Certain keywords do not require a value, and you may not provide one.

Required
A value must be provided if the keyword coded is designated as requiring a value.

Optional
Specification of a value is optional for some keywords.

Single
Only one value may be provided for some keywords.

Multiple
For some keywords, more than one value is permitted. (In some cases, DYNALLOC requires more than one value, although the dynamic allocation sample program does not enforce this).

Character string
Any characters are permitted in this type of value, although in most cases there will be additional rules to follow, for example, for the DSNAME keyword.

Numeric string
Only numeric characters are allowed for this type of value, for example, for the EXPDT keyword.

Maximum and minimum lengths
For character and numeric values, the maximum and minimum lengths of the value are checked by the program. For a fixed length string, these values are the same. The value will still be passed to DYNALLOC as specified.

Convertible to n byte binary
A numeric value is required, of a magnitude representable in binary in the specified number of bytes. Values that are too large are truncated to the maximum possible for the width.

The dynamic allocation sample program does not support negative numbers. It does not cross-check operand keywords; errors of this type will usually cause DYNALLOC to return error codes of the form 03xx.

Abbreviation rules for keywords

Keywords may be abbreviated. A word in the command matches a keyword if:

1. The spelling is the same, or
2. The first letter is the same, and the remaining letters in the word appear in the same order as they do in the keyword.

If an ambiguity occurs, the program diagnoses the ambiguity, and lists the possible keywords.

System programming considerations

Keyword spellings are defined in the program's table, DFH99T, which is link-edited with the program. Where possible, these are the same as the corresponding job control or TSO keywords. Comments in the source code for DFH99T explain how the system programmer may:

- Change the spelling of keywords
- Define alternative spelling for keywords
- Divide the functions of a verb into subsets
- Add new verbs with subset function
- Add new operands as they become available in the SVC.

Member DFH99BLD in CICS212.JCLLIB is the job stream used to build the program. If part of the program has been modified, reassemble that part and link-edit the program again.

The macro instructions IEFZB4D0 (DYNALLOC parameter list structure) and IEFZB4D2 (symbolic key equates), provided by MVS, are used in the dynamic allocation program and its keyword table. The meaning of each keyword in the table is defined in terms of a symbolic name, defined by one of the macros IEFZB4D0 or IEFZB4D2. These symbolic names are also documented in *OS/VS2 MVS System Programming Library: Job Management* manual. The definitions of command keywords given in that manual should be regarded in preference to those in any other source. To obtain a list of command keywords and their symbolic values, for use as a cross-reference to the MVS manual, assemble DFH99T with option SYSPARM(LIST), and print the resulting object code. If the table is changed, repeat the assembly to obtain a new list.

Chapter 6.3. Loading and accessing files that use phonetic codes for keys

General-Use Programming Interface

This chapter explains how the DFHPHN macro instruction is used, and should be read in conjunction with the section on built-in functions in the *CICS/VS Application Programmer's Reference Manual (Macro Level).*

The major use of phonetic codes is for keys to data sets. In this way, records can be accessed even if a key is misspelled. The phonetic code conversion subroutine (DFHPHN) is provided to assist you in loading and accessing such data sets offline. DFHPHN is generated by specifying the built-in functions program DFHSG PROGRAM=BFP.

This offline subroutine enables you to convert a 16-character name to a four-byte phonetic code. See the built-in function macro instruction (DFHBIF TYPE=PHONETIC) in the *CICS/VS Application Programmer's Reference Manual (Macro Level)* for the rules of the conversion.

This function can be invoked by a program running under any of the operating systems under which CICS can be run. The calling format is:

```
CALL    DFHPHN,(lang,name,phon)          assembler
CALL    DFHPHN (lang,name,phon)          PL/I
CALL    'DFHPHN' USING lang name phon    COBOL
```

where:

lang
is the symbolic address of a field that contains a one-byte language indicator.

If an error occurs during processing of this request, X'50' is returned in this location. If no error occurs, X'00' is returned and the location must be reset to indicate the programming language before the location can be reused.

X'F0'	indicates assembler or COBOL
X'F1'	indicates PL/I

name
is the symbolic address of a field that contains the 16-character name.

phon
is the symbolic address of a field in which the four-byte phonetic code is returned. If the first character of the "name" field is not alphabetic, the "lang" field will be set to X'50'.

The steps in loading such a data set would typically be:

1. Create the keys.
   a. Read a record from the source data set
   b. Generate the code using a call to the DFHPHN subroutine
   c. Write the record on a temporary sequential data set.
2. Sort the temporary data set on phonetic code.
3. Load the key-sequenced VSAM data set
   a. Read the sorted temporary data set
   b. Write to the keyed data set.

End of General-Use Programming Interface

Appendixes

Appendix A. Program generation summary

Diagnosis, Modification and Tuning Information

This appendix contains two lists of the modules that you can generate using DFHSG TYPE=INITIAL and DFHSG PROGRAM=xxx macros. The lists are in alphabetic order. The first list is ordered on the names of the modules generated, and the second on the DFHSG program group names.

The superscript numbers refer to the footnotes at the end of the lists. There are notes about some of the modules generated under the descriptions of individual commands in "Chapter 1.2. DFHSG PROGRAM=xxx" on page 13.

System generation modules listed by module name

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHACEE	Security ID program	CSS
DFHACP	Abnormal condition	CSO
DFHAKP	Activity key point	KPP
DFHALP	Allocation program (terminal resources)	KCP
DFHAMP	RDO allocation manager program	CSO
DFHASV	Page fix/free SVC routine	INITIAL
DFHBFP[1]	Built-in function	BFP
DFHBRCP	CBRC transaction	EIP
DFHCAA70[2]	7770 channel appendage program	CSO
DFHCAP	Utility command analyzer program	CSO
DFHCCMF	Periodic monitoring program	CSO
DFHCICS	Service level indicator	CSA
DFHCMON	Start/stop monitoring program	CSO
DFHCMP[1]	Monitoring program	CSO
DFHCPY	VTAM 3270 print function support	TCP
DFHCRC	IRC CICS STAE exit	CSO
DFHCRNP	Interregion connection manager	ISC
DFHCRP	Relay program	ISC
DFHCRQ	ATI purge program	ISC
DFHCRR	Interregion recovery	ISC
DFHCRS	Remote scheduler	ISC
DFHCRSP	Interregion control initialization module	ISC
DFHCSA	Common systems area	CSA
DFHCSDUP	RDO offline utility program	CSU
DFHCSVC[3]	Bootstrap type 2 SVC	INITIAL
DFHCURDI	Standard definitions for CSD initialize	CSU
DFHCUS1B	CSD service utility program	CSU
DFHCU17S	Sample definitions for CSD initialize	CSU
DFHCU17T	Resource definition list for CSD	CSU
DFHCU170	Standard definitions for CSD upgrade	CSU
DFHCWTO	Console-write-to-operator	CSO
DFHDBMS	Temporary storage browse mapset	EIP
DFHDBP	Dynamic transaction backout	DBP
DFHDCP	Dump control	DCP
DFHDEB70	7770 device end program	CSO

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHDES	Data encryption standard	TCP
DFHDIP[1]	Batch data interchange program	DIP
DFHDLBP	DL/I backout program	TBP
DFHDLG	DL/I global command processor	CSO
DFHDLI[1]	DL/I interface	CSO
DFHDLIAI	DL/I application interface stub	EIP
DFHDLQ	"IMS/VS" quasi-application program	CSO
DFHDLR	IMS/VS simulated routines	CSO
DFHDLRP	DL/I restart program	CSO
DFHDLS	DL/I status processor	CSO
DFHDLX	IMS/VS internal routines	CSO
DFHDMP	RDO CICS system definition file (CSD) manager	CSO
DFHDRP	DL/I shared database bootstrap program	ISC
DFHDRP(A-G)	DL/I shared database batch modules	ISC
DFHDSB	Data stream builder	BMS
DFHDUP	Dump utility	CSU
DFHEAI	Assembler EXEC link-edit stub	EXP
DFHEAI0	Assembler EXEC link-edit stub	EXP
DFHEAP1$	Assembler EXEC interface translator	EXP
DFHEBF	EXEC BFP module	EIP
DFHEBRCT	Table for CBRC transaction	EIP
DFHEBU	ISC (FMH building)	EIP
DFHECI	COBOL EXEC interface link-edit stub	EXP
DFHECID	Command interpreter	EIP
DFHECIP	CECI initialization	EIP
DFHECP1$	COBOL EXEC interface translator	EXP
DFHECSP	CECS initialization	EIP
DFHEDAD	CEDA module	EIP
DFHEDAP	CEDA initialization	EIP
DFHEDC	EXEC DCP module	EIP
DFHEDFBR	Temporary storage browse program	EIP
DFHEDFD	EDF display program	EIP
DFHEDFM	EDF map set	EIP
DFHEDFP	EDF control program	EIP
DFHEDFR	EDF response table	EIP
DFHEDFX	EDF task switch program	EIP
DFHEDI	EXEC DIP module	EIP
DFHEDP	EXEC DL/I processor	CSO
DFHEEI	EXEC EIP module	EIP
DFHEEX	ISC (FMH extraction)	EIP
DFHEFC	EXEC FCP module	EIP
DFHEGL	EXEC LU6.2	EIP
DFHEIC	EXEC ICP module	EIP
DFHEIDLI	EXEC DL/I translator module	EXP
DFHEIGDS	EXEC GDS table module	EXP
DFHEIGDX	EXEC GDS translator module	EXP
DFHEIP	EXEC interface program	EIP
DFHEITAB	Translator table	EIP
DFHEITCU	RDO offline utility language table	CSU
DFHEITMT	CEMT language table	EIP
DFHEITOT	CEOT language table	EIP
DFHEITSP	CEDA language table	EIP
DFHEITST	CEST language table	EIP
DFHEJC	EXEC JCP module	EIP
DFHEKC	EXEC KCP module	EIP

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHELR	EXEC local/remote module	ISC
DFHEMA	Master terminal program	EIP
DFHEMB	Master terminal program	EIP
DFHEMC	Master terminal program	EIP
DFHEMD	Master terminal program	EIP
DFHEME	Master terminal program	EIP
DFHEMF	Master terminal program	EIP
DFHEMG	Master terminal program	EIP
DFHEMH	Master terminal program	EIP
DFHEMI	Master terminal program	EIP
DFHEMS	EXEC BMS module	EIP
DFHEMTA	Master terminal control module	EIP
DFHEMTD	Enhanced master terminal module	EIP
DFHEMTP	CEMT initialization	EIP
DFHEOTP	CEOT initialization	EIP
DFHEPC	EXEC PCP module	EIP
DFHEPI	PL/I EXEC interface link-edit stub	EXP
DFHEPP1$	PL/I EXEC interface translator	EXP
DFHERM	Resource manager interface module	EIP
DFHESC	EXEC SCP module	EIP
DFHESP	EXEC SPP module	EIP
DFHESTP	CEST initialization	EIP
DFHETC	EXEC TCP/ZCP module	EIP
DFHETD	EXEC TDP module	EIP
DFHETL	EXEC LU6.2	EIP
DFHETR	EXEC TRP module	EIP
DFHETS	EXEC TSP module	EIP
DFHEXI	VTAM 3270 print function support	TCP
DFHFCBP	File control backout program	TBP
DFHFCRP	File control recovery program	CSO
DFHFCU	File control utility	CSO
DFHFDP	Formatted dump program	CSO
DFHFEP	FE terminal test program	CSS
DFHFTAP	Format tape program	KPP
DFHGAP	Graphics attention program	GAP
DFHGMM	VTAM Good Morning message program	TCP
DFHHPSVC	Service request block type 6 supervisor	INITIAL
DFHICP	Interval control program	ICP
DFHIIP	Non-3270 input mapping	BMS
DFHIRP	Interregion control program	ISC
DFHISP	Intercommunication program	ISC
DFHJCBSP	Journal tasks bootstrap program	JCP
DFHJCC	Journal control close	JCP
DFHJCEOV	Journal control EOV	JCP
DFHJCI	Journal control input	JCP
DFHJCIOE	Journal control I/O error program	JCP
DFHJCJFP	Journal control format program	JCP
DFHJCKOJ	Journal control kickoff program	JCP
DFHJCO	Journal control open	JCP
DFHJCOCP	Journal control open/close program	JCP
DFHJCP[1]	Journal control	JCP
DFHJCSDJ	Journal control shutdown	JCP
DFHKCP	Task control	KCP
DFHKCRP	Task control restart program	CSO
DFHKCSP	SRB service program	KCP

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHLFO	LIFO storage program	CSO
DFHLUP	LU services manager program	ISC
DFHMCP	Mapping control	BMS
DFHMCX	Fast path module	BMS
DFHMGP	Error message program	CSO
DFHMGT	Error message table	CSO
DFHMIR	ISC mirror program	ISC
DFHML1	LU1 printer mapping	BMS
DFHMSP	Message switching program	CSO
DFHMTPA	Master terminal program module A	MTP
DFHMTPB	Master terminal program module B	MTP
DFHMTPC	Master terminal program module C	MTP
DFHMTPD	Master terminal program module D	MTP
DFHMTPE	Master terminal program module E	MTP
DFHMTPF	Master terminal program module F	MTP
DFHMTPG	Master terminal program module G	MTP
DFHMXP	Local queuing shipper program	ISC
DFHM32	BMS 3270 mapping	BMS
DFHPBP	BMS page build program	BMS
DFHPCP	Program control	PCP
DFHPCRP	Program control restart program	CSO
DFHPEP	Program error dummy program	CSO
DFHPHN	Phonetic code conversion program	BFP
DFHPHP	Partition handling program	BMS
DFHPRK	VTAM 3270 print function support	TCP
DFHPRPR	HLL preprocessor	HLL
DFHPUP	RDO parameter utility program	CSO
DFHPXR	Subtask post exit routine	CSO
DFHP3270	3270 print function support	TCP
DFHRCEX	Recovery control enable exits	CSO
DFHRCRP	Recovery control restart program	CSO
DFHRKB	VTAM 3270 print function support	TCP
DFHRLR	BMS route list resolution	BMS
DFHRMSY	Resource manager	KPP
DFHRTE	Transaction routing program	ISC
DFHRTY	CICS-supplied transaction restart module	DBP
DFHRUP	Recovery utility program	KPP
DFHRWP70	7770 read/write	CSO
DFHSCP	Storage control	SCP
DFHSCR	Storage control recovery	SCP
DFHSFP	Sign-off program	CSS
DFHSIA1	System initialization – module A1	CSO
DFHSIB1	System initialization – module B1	CSO
DFHSIC1	System initialization – module C1	CSO
DFHSID1	System initialization – module D1	CSO
DFHSIE1	System initialization – module E1	CSO
DFHSIF1	System initialization – module F1	CSO
DFHSIG1	System initialization – module G1	CSO
DFHSIH1	System initialization – module H1	CSO
DFHSII1	System initialization – module I1	CSO
DFHSIJ1	System initialization – module J1	CSO
DFHSIP	System initialization	CSO
DFHSNP	Sign-on program	CSS
DFHSPP	Sync point program	KCP
DFHSPZ	ISC syncpoint protocol program	KCP

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHSRP	System recovery	SRP
DFHSTKC	Supervisor statistics	CSO
DFHSTLK	ISC link statistics program	CSO
DFHSTP	System termination	CSO
DFHSTPD	Program and dump statistics	CSO
DFHSTSP	Auto. statistics summarization control	CSO
DFHSTTD	Data management statistics	CSO
DFHSTTR	File and terminal statistics	CSO
DFHSTUP	Auto. statistics summarization utility	CSU
DFHTACP	Terminal abnormal condition	CSO
DFHTAJP	Time of day adjustment	CSO
DFHTCP	Terminal control program	TCP
DFHTCBP	Terminal control backout program	TBP
DFHTDP	Transient data program	TDP
DFHTDRP	Transient data recovery program	KPP
DFHTEOF	Tape end of file program	KPP
DFHTEP	Sample terminal error program	CSO
DFHTEPT	Sample terminal error table	CSO
DFHTMP	Table management program	CSO
DFHTPP	BMS terminal page program	BMS
DFHTPQ	BMS terminal page clean-up	BMS
DFHTPR	BMS terminal page retrieval	BMS
DFHTPS	BMS delayed message delivery	BMS
DFHTRAP	FE global trap exit program	CSS
DFHTRP	Trace program	TRP
DFHTSBP	Temporary storage backout program	TBP
DFHTSP	Temporary storage program	TSP
DFHTSRP	Temporary storage recovery program	KPP
DFHTUP	Trace utility program	CSU
DFHUEH	User exit handler	CSO
DFHUEM	User exit manager	CSO
DFHUSBP	User backout program	TBP
DFHVAP	VSAM subtask monitor program	CSO
DFHVCP[1]	Volume control manager	JCP
DFHVSP	VSAM subtask program	CSO
DFHWKP	Warm keypoint program	CSO
DFHXFP	ISC transformer program	ISC
DFHXFQ	DL/I shared database transformer program	ISC
DFHXFX	Fast-path transformer program for MRO function shipping	ISC
DFHXMP	Interregion switch routine	ISC
DFHXSP	Security program	CSS
DFHXSS	SVC link module to RACF	CSS
DFHXTP	Transaction routing transformer	ISC
DFHZCA	VTAM terminal control program module	TCP
DFHZCB	VTAM terminal control program module	TCP
DFHZCC	VTAM terminal control program module	TCP
DFHZCP	Common terminal control program module	TCP
DFHZCW	VTAM terminal control program module	TCP
DFHZCX	Common terminal control program module	TCP
DFHZCY	VTAM terminal control program module	TCP
DFHZCZ	VTAM terminal control program module	TCP
DFHZHPRX	RPL executor in SRB Mode	TCP
DFHZNAC	Node abnormal condition program	TCP
DFHZNEP	Node error program interface program	TCP

MODULE NAME	MODULE DESCRIPTION	DFHSG PROGRAM=
DFHZRLG	Response logging program	TCP
DFHZRSP	Resend program	TCP

System generation modules listed by DFHSG PROGRAM=

DFHSG PROGRAM=	MODULE NAME	MODULE DESCRIPTION
INITIAL	DFHASV	Page fix/free SVC routine
	DFHCSVC[3]	Bootstrap type 2 SVC
	DFHHPSVC	Service request block type 6 supervisor
BFP	DFHBFP[1]	Built-in function
	DFHPHN	Phonetic code conversion program
BMS	DFHDSB	Data stream builder
	DFHIIP	Non-3270 input mapping
	DFHMCP	Mapping control
	DFHMCX	Fast path module
	DFHML1	LU1 printer mapping
	DFHM32	BMS 3270 mapping
	DFHPBP	BMS page build program
	DFHPHP	Partition handling program
	DFHRLR	BMS route list resolution
	DFHTPP	BMS terminal page program
	DFHTPQ	BMS terminal page clean-up
	DFHTPR	BMS terminal page retrieval
	DFHTPS	BMS Delayed Message delivery
CSA	DFHCICS	Service level indicator
	DFHCSA	Common systems area
CSO	DFHACP	Abnormal condition
	DFHAMP	RDO allocation manager program
	DFHCAA70[2]	7770 channel appendage program
	DFHCAP	Utility command analyzer program
	DFHCCMF	Periodic monitoring program
	DFHCMON	Start/stop monitoring program
	DFHCMP[1]	Monitoring program
	DFHCRC	IRC &cics. STAE exit
	DFHCWTO	Console-write-to-operator
	DFHDEB70	7770 device end program
	DFHDLG	DL/I global command processor
	DFHDLI[1]	DL/I Interface
	DFHDLQ	"IMS/VS" quasi-application program
	DFHDLR	IMS/VS simulated routines
	DFHDLRP	DL/I restart program
	DFHDLS	DL/I status processor
	DFHDLX	IMS/VS internal routines
	DFHDMP	RDO CICS system definition file (CSD) manager
	DFHEDP	EXEC DL/I processor
	DFHFCRP	File control recovery program
	DFHFCU	File control utility
	DFHFDP	Formatted dump program
	DFHKCRP	Task control restart program
	DFHLFO	LIFO storage program
	DFHMGP	Error message program
	DFHMGT	Error message table
	DFHMSP	Message switching program
	DFHPCRP	Program control restart program
	DFHPEP	Program error dummy program
	DFHPUP	RDO parameter utility program
	DFHPXR	Subtask post exit routine
	DFHRCEX	Recovery control enable exits

DFHSG PROGRAM=	MODULE NAME	MODULE DESCRIPTION
	DFHRCRP	Recovery control restart program
	DFHRWP70	7770 read/write
	DFHSIA1	System initialization – module A1
	DFHSIB1	System initialization – module B1
	DFHSIC1	System initialization – module C1
	DFHSID1	System initialization – module D1
	DFHSIE1	System initialization – module E1
	DFHSIF1	System initialization – module F1
	DFHSIG1	System initialization – module G1
	DFHSIH1	System initialization – module H1
	DFHSII1	System initialization – module I1
	DFHSIJ1	System initialization – module J1
	DFHSIP	System initialization
	DFHSTKC	Supervisor statistics
	DFHSTLK	ISC link statistics program
	DFHSTP	System termination
	DFHSTPD	Program and dump statistics
	DFHSTSP	Auto. statistics summarization control
	DFHSTTD	Data management statistics
	DFHSTTR	File and terminal statistics
	DFHTACP	Terminal abnormal condition
	DFHTAJP	Time of day adjustment
	DFHTEP	Sample terminal error program
	DFHTEPT	Sample terminal error table
	DFHTMP	Table management program
	DFHUEH	User exit handler
	DFHUEM	User exit manager
	DFHVAP	VSAM subtask monitor program
	DFHVSP	VSAM subtask program
	DFHWKP	Warm keypoint program
CSS	DFHACEE	Security ID program
	DFHFEP	FE terminal test program
	DFHSFP	Sign-off program
	DFHSNP	Sign-on program
	DFHTRAP	FE global trap exit program
	DFHXSP	Security program
	DFHXSS	SVC link module to RACF
CSU	DFHCSDUP	RDO offline utility program
	DFHCURDI	Standard definitions for CSD initialize
	DFHCUS1B	CSD service utility program
	DFHCU17S	Sample definitions for CSD initialize
	DFHCU17T	Resource definition list for CSD
	DFHCU170	Standard definitions for CSD upgrade
	DFHDUP	Dump utility
	DFHEITCU	RDO offline utility language table
	DFHSTUP	Auto. statistics summarization utility
	DFHTUP	Trace utility program
DBP	DFHDBP	Dynamic transaction backout
	DFHRTY	CICS-supplied transaction restart module
DCP	DFHDCP	Dump control
DIP	DFHDIP[1]	Batch data interchange program
EIP	DFHBRCP	CBRC transaction
	DFHDBMS	Temporary storage browse mapset
	DFHDLIAI	DL/I application interface stub
	DFHEBF	EXEC BFP module

DFHSG PROGRAM=	MODULE NAME	MODULE DESCRIPTION
	DFHEBRCT	Table for CBRC transaction
	DFHEBU	ISC (FMH building)
	DFHECID	Command interpreter
	DFHECIP	CECI initialization
	DFHECSP	CECS initialization
	DFHEDAD	CEDA module
	DFHEDAP	CEDA initialization
	DFHEDC	EXEC DCP module
	DFHEDFBR	Temporary storage browse program
	DFHEDFD	EDF display program
	DFHEDFM	EDF map set
	DFHEDFP	EDF control program
	DFHEDFR	EDF response table
	DFHEDFX	EDF task switch program
	DFHEDI	EXEC DIP module
	DFHEEI	EXEC EIP module
	DFHEEX	ISC (FMH extraction)
	DFHEFC	EXEC FCP module
	DFHEGL	EXEC LU6.2
	DFHEIC	EXEC ICP module
	DFHEIP	EXEC interface program
	DFHEITAB	Translator table
	DFHEITMT	CEMT language table
	DFHEITOT	CEOT language table
	DFHEITSP	CEDA language table
	DFHEITST	CEST language table
	DFHEJC	EXEC JCP module
	DFHEKC	EXEC KCP module
	DFHEMA	Master terminal program
	DFHEMB	Master terminal program
	DFHEMC	Master terminal program
	DFHEMD	Master terminal program
	DFHEME	Master terminal program
	DFHEMF	Master terminal program
	DFHEMG	Master terminal program
	DFHEMH	Master terminal program
	DFHEMI	Master terminal program
	DFHEMS	EXEC BMS module
	DFHEMTA	Master terminal control module
	DFHEMTD	Enhanced master terminal module
	DFHEMTP	CEMT initialization
	DFHEOTP	CEOT initialization
	DFHEPC	EXEC PCP module
	DFHERM	Resource manager interface module
	DFHESC	EXEC SCP module
	DFHESP	EXEC SPP module
	DFHESTP	CEST initialization
	DFHETC	EXEC TCP/ZCP module
	DFHETD	EXEC TDP module
	DFHETL	EXEC LU6.2
	DFHETR	EXEC TRP module
	DFHETS	EXEC TSP module
EXP	DFHEAI	Assembler EXEC link-edit stub
	DFHEAI0	Assembler EXEC link-edit stub
	DFHEAP1$	Assembler EXEC Interface Translator

DFHSG PROGRAM=	MODULE NAME	MODULE DESCRIPTION
	DFHECI	COBOL EXEC interface link-edit stub
	DFHECP1$	COBOL EXEC interface translator
	DFHEIDLI	EXEC DL/I translator module
	DFHEIGDS	EXEC GDS table module
	DFHEIGDX	EXEC GDS translator module
	DFHEPI	PL/I EXEC interface link-edit stub
	DFHEPP1$	PL/I EXEC interface translator
GAP	DFHGAP	Graphics attention program
HLL	DFHPRPR	HLL preprocessor
ICP	DFHICP	Interval control program
ISC	DFHCRNP	Interregion connection manager
	DFHCRP	Relay program
	DFHCRQ	ATI purge program
	DFHCRR	Interregion recovery
	DFHCRS	Remote scheduler
	DFHCRSP	Interregion control initialization module
	DFHDRP	DL/I shared database bootstrap program
	DFHDRP(A-G)	DL/I shared database batch modules
	DFHELR	EXEC local/remote module
	DFHIRP	Interregion control program
	DFHISP	Intercommunication program
	DFHLUP	LU services manager program
	DFHMIR	ISC mirror program
	DFHMXP	Local queuing shipper program
	DFHRTE	Transaction routing program
	DFHXFP	ISC transformer program
	DFHXFQ	DL/I shared database transformer program
	DFHXFX	Fast-path transformer program for MRO function shipping
	DFHXMP	Interregion switch routine
	DFHXTP	Transaction routing transformer
JCP	DFHJCBSP	Journal tasks bootstrap program
	DFHJCC	Journal control close
	DFHJCEOV	Journal control EOV
	DFHJCI	Journal control input
	DFHJCIOE	Journal control I/O error program
	DFHJCJFP	Journal control format program
	DFHJCKOJ	Journal control kickoff program
	DFHJCO	Journal control open
	DFHJCOCP	Journal control open/close program
	DFHJCP[1]	Journal control
	DFHJCSDJ	Journal control shutdown
	DFHVCP[1]	Volume control manager
KCP	DFHALP	Allocation program (terminal resources)
	DFHKCP	Task control
	DFHKCSP	SRB service program
	DFHSPP	Sync point program
	DFHSPZ	ISC syncpoint protocol program
KPP	DFHAKP	Activity key point
	DFHFTAP	Format tape program
	DFHRMSY	Resource manager
	DFHRUP	Recovery utility program
	DFHTDRP	Transient data recovery program
	DFHTEOF	Tape End of file program
	DFHTSRP	Temporary storage recovery program

DFHSG PROGRAM=	MODULE NAME	MODULE DESCRIPTION
MTP	DFHMTPA	Master terminal program module A
	DFHMTPB	Master terminal program module B
	DFHMTPC	Master terminal program module C
	DFHMTPD	Master terminal program module D
	DFHMTPE	Master terminal program module E
	DFHMTPF	Master terminal program module F
	DFHMTPG	Master terminal program module G
PCP	DFHPCP	Program control
SCP	DFHSCP	Storage control
	DFHSCR	Storage control recovery
SRP	DFHSRP	System recovery
TBP	DFHDLBP	DL/I backout program
	DFHFCBP	File control backout program
	DFHTCBP	Terminal control backout program
	DFHTSBP	Temporary storage backout program
	DFHUSBP	User backout program
TCP	DFHCPY	VTAM 3270 print function support
	DFHDES	Data encryption standard
	DFHEXI	VTAM 3270 print function support
	DFHGMM	VTAM Good Morning message program
	DFHPRK	VTAM 3270 print function support
	DFHP3270	3270 print function support
	DFHRKB	VTAM 3270 print function support
	DFHTCP	Terminal control program
	DFHZCA	VTAM terminal control program module
	DFHZCB	VTAM terminal control program module
	DFHZCC	VTAM terminal control program module
	DFHZCP	Common terminal control program module
	DFHZCW	VTAM terminal control program module
	DFHZCX	Common terminal control program module
	DFHZCY	VTAM terminal control program module
	DFHZCZ	VTAM terminal control program module
	DFHZHPRX	RPL executor in SRB Mode
	DFHZNAC	Node abnormal condition program
	DFHZNEP	Node error program interface program
	DFHZRLG	Response logging program
	DFHZRSP	Resend program
TDP	DFHTDP	Transient data program
TRP	DFHTRP	Trace program
TSP	DFHTSP	Temporary storage program

[1] The corresponding dummy program is supplied without the need for SYSGEN.

[2] Link edited as IGG019xx, where xx is the appendage suffix specified in the CAA operand of the DFHSG PROGRAM=CSO macro.

[3] Link edited into nucleus as IGCxxx, where xxx is the value given to CICSSVC.

Appendix B. Sample TCAM SNA message control programs

This appendix contains two sample TCAM SNA message control programs (MCPs). The MCPs given have the following functions:

Sample 1	Control of the SNA sessions independently of the CICS application programs
Sample 2	Controls the SNA sessions according to the requirements of the CICS application programs.

Further information on TCAM devices in a TCAM SNA environment and on MCPs for TCAM SNA devices is given in "Chapter 4.2. The CICS/TCAM interface" on page 205 under "CICS with TCAM SNA."

The statements listed are those of the sample programs supplied with the current release of CICS. Sample programs shipped with subsequent PTFs may differ from these listings.

SAMPLE 1: DFHSPTM1 — SAMPLE TCAM MCP FOR TCAM DIRECT

```
***********************************************************************
*
*
*
*  NAME - CICS SAMPLE TCAM MESSAGE CONTROL PROGRAM AND MESSAGE HANDLERS
*         (INDEPENDENT CONTROL)
*
*  PURPOSE - THE PURPOSE OF THIS SAMPLE IS TO DEMONSTRATE TO THE CICS
*        USER WHAT IS REQUIRED FOR THE CREATION OF A SIMPLE CICS SNA
*        NETWORK USING TCAM.
THE SAMPLE DEMONSTRATES HALF-DUPLEX
*        FLIP/FLOP MODE WITH BRACKETS.
*
*
*  FUNCTIONS - THE FUNCTIONS ARE AS FOLLOWS:
*
*     INTRODUCTION - DEFINE THE OVERALL SYSTEM PARAMETERS, INITIALIZE
*        THE SYSTEM, AND START MESSAGE TRAFFIC.
*
*     DEFINITION - DESCRIBE THE SPECIFICS OF THE NETWORK, THE MESSAGE
*        QUEUES, AND THE NECESSARY CONTROL BLOCKS.
*
*     DEVICE MESSAGE HANDLER - INSERT THE COMMUNICATIONS CONTROL BYTES
*        AND ROUTE THE MESSAGE FOR INPUT,  REMOVE THE CCB AND ROUTE
*        THE MESSAGE TO ITS PROPER DESTINATION ON OUTPUT.
THE DMH IS
*        NAMED 'CICS' SO THAT AN LU CAN LOGON TO 'CICS'.
*        THIS MH SUPPORTS LU TYPE0, TYPE1, AND TYPE2. THE SUPPORT IS
*        DESIGNED TO ALLOW THE DEVICE MH TO CONTROL THE LU.
*        LU TYPE 1 SUPPORT:
*        THE LU MUST BE BOUND TO ALLOW IT TO SEND END BRACKET. THE HOST
*        WILL BEGIN AND END A BRACKET ON EVERY CHAIN EXCEPT WHEN A
*        DATASET IS BEING SENT.
```

```
IN THIS CASE THE BRACKET WILL NOT BE
*        ENDED UNTIL THE END OF DATASET. OTHER METHODS OF OPERATION
*        ARE POSSIBLE BY USING DIFFERENT MH OPTIONS.
THE LU TYPE 1
*        BATCH SUPPORT ASSUMES A SINGLE TRANSACTION WILL HANDLE THE
*        BATCH INPUT.
THE MH WILL EDIT THE TRANSACTION NAME INTO
*        THE FIRST CHAIN OF THE DATASET.
THEREFORE IT IS NOT NECESSARY
*        TO PLACE A TRANSACTION NAME INTO THE DATASET.
*        LU TYPE 2 SUPPORT:
*        LU TYPE 2 IS THE 3270 DATA STREAM EMULATOR.
*        THE SUPPORT IS DESIGNED TO ALLOW THE TRANSACTION TO OPERATE
*        INDEPENDENTLY FROM THE OUTBOARD LU.
THEREFORE THE KEYBOARD
*        IS UNLOCKED AFTER EVERY INPUT.
SINCE AN LU TYPE 2 CANNOT BE
*        BOUND TO SEND END BRACKET A MSGGEN IS USED TO UNLOCK IT.
*        ALSO THE HOST WILL BEGIN AND END A BRACKET ON EVERY CHAIN.
*        THIS WILL CAUSE THE KEYBOARD TO UNLOCK AFTER EVERY OUTPUT
*        MESSAGE.
IF A DIFFERENT METHOD OF OPERATION IS DESIRED
*        LOGIC COULD BE ADDED TO ONLY END THE BRACKET WHEN THE KEYBOARD
*        UNLOCK SEQUENCE IS SENT.
*
*
*     APPLICATION MESSAGE HANDLER - ROUTE MESSAGES FROM THE INPUT QUEUE
*        TO CICS AND FROM CICS TO THE APPROPRIATE OUTPUT QUEUE.
*
*
*     THE SSCP MESSAGE HANDLER  - UTILIZES THE IBM-SUPPLIED MH  TO
*        PERFORM THE NECESSARY ROUTING AND ANALYSIS FUNCTIONS.
COMPLEX
*        USER SYSTEMS MAY REQUIRE THIS TO BE MODIFIED BY THE USER.
*
*
*  NOTES -
*
*     CONVENTIONS -
*
*        REGISTER 2 IS USED AS THE DCB REGISTER
*
*        REGISTER 3 USED AS INTERNAL LINKAGE REGISTER
*
*        REGISTER 4 USED AS INTERNAL WORK REGISTER
*
*        REGISTER 5 USED AS INTERNAL WORK REGISTER
*
*        REGISTER 6 USED AS THE SCAN POINTER REGISTER
*
*     DEFAULTS -
*
*        MACRO DEFAULTS ARE USED WHEREVER REASONABLE
*
```

```
*
*     EXITS -
*
*        NORMAL -
*
*        RETURN TO THE MVS SUPERVISOR WHEN SHUTDOWN IS COMPLETE
*
*     ERROR -
*
*        X'FFF' - ABEND ON INTRO FAILURE
*
*        X'FFE' - ABEND ON MESSAGE QUEUE DCB OPEN FAILURE
*
*        X'FFD' - ABEND ON 3705 DCB OPEN FAILURE
*
*
*
*
***********************************************************************
CICSTCAM CSECT
RDCB     EQU   2                        DCB REGISTER
LINKREG  EQU   3                        INTERNAL LINKAGE REGISTER
RWORK    EQU   4                        INTERNAL WORK REGISTER
RSCANSVE EQU   5                        SAVED SCAN POINTER REGISTER
RSCAN    EQU   6                        SCAN POINTER REGISTER
RRETURN  EQU   15
OPEN     EQU   X'10'                    DCB OPEN FLAG
DCBOFLGS EQU   X'30'                    OPEN FLAGS OFFSETF
WORD     EQU   4                        OFFSET
         SPACE 2
*        CCB BYTE 0
         SPACE
CCBFMH   EQU   X'01'                    FORMATTED HEADER
CCBDISC  EQU   X'08'                    DISCONNECT
         SPACE 2
*        CCB BYTE 1
         SPACE
CCBEB    EQU   X'01'                    WRITE LAST SPECIFIED
CCBCD    EQU   X'40'                    WRITE WITH READ SPECIFIED
         SPACE 2
FMHLEN   EQU   0                        BYTE 0 OF FMH
FMHTYPE  EQU   1                        BYTE 1 OF FMH
FMHSEL   EQU   2                        BYTE 2 OF FMH
FMHSTCK  EQU   3                        BYTE 3 OF FMH
FMHPROP  EQU   4                        BYTE 4 OF FMH
FMHTYP1  EQU   X'01'                    TYPE 1 FMH
FMHBDS   EQU   X'40'                    BEGIN DATASET FMH
FMHEDS   EQU   X'20'                    END DATASET FMH
PRFSTAT1 EQU   X'14'                    STATUS BYTE OFFSET
PRFNLSTN EQU   X'02'                    NOT LAST INDICATOR
ZERO     EQU   0
ONE      EQU   1
TWO      EQU   2
FOUR     EQU   4
**********************************************************************
*        SNACTL OPTION FIELD USAGE
```

```
SNARCD   EQU   X'01'                  REMEMBER TO SET CD
SNASCD   EQU   X'02'                  CD HAS BEEN SENT TCAM CANNOT
*                                     SEND ANY DATA
SNASDS   EQU   X'04'                  SEND DATASET STATE
SNARDS   EQU   X'08'                  RECEIVE DATASET STATE
SNALUT2  EQU   X'80'                  TYPE 2 LU THE 3270 DSE
*********************************************************************
         EJECT
         INTRO PROGID=CICSTCAM,                                        X
               UNITSZ=160,            BUFFER UNIT SIZE                 X
               LNUNITS=100,           SEE TCAM SYSTEM PGMER'S GUIDE  X
               BRACKET=YES,           INCLUDE BRACKET STATE MANAGER  X
               BTRACE=500,            PIU TRACE ENTRIES                X
               CIB=5,                 MAX 5 ACTIVE OPERATOR COMMANDS X
               COMWRTE=YES,           INCLUDE SERVICE AID WRITER       X
               CONTROL=OPCTL,         ALLOW COMMANDS FROM OTHERS       X
               CPB=20,                ACTIVE DISK CHANNEL PROGRAMS     X
               DISK=YES,              DISK MESSAGE QUEUES              X
               DLQ=0,                 NO DEAD LETTER QUEUE             X
               DTRACE=500,            DISPATCHER TRACE ENTRIES         X
               FEATURE=(NODIAL,NO2741,,,ONLY3705,ONLYSNA),             X
               MSUNITS=100,           BUFFER UNITS FOR CORE QUEUE      X
               MAXSUBA=3,             OBTAIN FROM NCP GENERATION       X
               PLCBNO=20,             PSEUDO LCBS NEEDED FOR I/O       X
               PRIMARY=SYSCON,        OPERATOR CONTROL CONSOLE         X
               SIBCNT=25,             MAXIMUM SNA SESSIONS PERMITTED X
               SUBAREA=1              OBTAIN FROM NCP GENERATION
         LTR   RRETURN,RRETURN        WAS INTRO SUCCESSFUL
         BZ    OKINTRO                IF SO, CARRY ON
         STH   RRETURN,DEBUG          MAKE RETURN CODE VISIBLE
         ABEND 4095,DUMP              OTHERWISE, PUNT WITH X'FFF'
OKINTRO  OPEN  (MSGQUEUE,(INOUT))     OPEN MESSAGE QUEUES DATA SET
         LA    RDCB,MSGQUEUE          POINT TO DCB
         TM    DCBOFLGS(RDCB),OPEN    WAS OPEN SUCCESSFUL
         BO    OKOPENQS               IF SO, CARRY ON
         ABEND 4094,DUMP              OTHERWISE, PUNT WITH X'FFE'
OKOPENQS OPEN  (NCP1DCB,(INOUT))      OPEN 3705 FOR COMMUNICATIONS
         LA    RDCB,NCP1DCB           POINT TO 3705 DCB
         TM    DCBOFLGS(RDCB),OPEN    WAS OPEN SUCCESSFUL
         BO    OKOPEN05               IF SO, CARRY ON
         ABEND 4093,DUMP              OTHERWISE, PUNT WITH X'FFD'
OKOPEN05 READY                        LET TCAM START TRAFFIC
         CLOSE (NCP1DCB,,MSGQUEUE,)
         RETURN     (14,12)           RELINQUISH CONTROL
         EJECT
***********************************************************************
*
*        DEFINE THE CONFIGURATION OF THE NETWORK - PHYSICAL AND LOGICAL
*
***********************************************************************
         SPACE 2
         DS    0D
         DC    C'  RETURN CODE = '    TRIGGER FOR DUMP SCANNING
DEBUG    DS    H                      TO CONTAIN INTRO RETURN CODE
         DC    C'  '                  SPACING AROUND MESSAGE
MSGQUEUE DCB   DSORG=TQ,              MESSAGE QUEUE DATA SET           X
```

```
                DDNAME=MSGQUEUE,                                       X
                MACRF=(G,P),                                           X
                OPTCD=R                 REUSABLE DISK QUEUES
NCP1DCB  DCB    DSORG=TR,               3705 COMMUNICATIONS CONTROLLER X
                DDNAME=DDNCP1,                                         X
                MACRF=(G,P)
CICSPCB  PCB    MH=AMH,BUFSIZE=2000     APPLICATION PROGRAM MH
         TTABLE LAST=SSCP               TERMINAL TABLE START AND END
SNACTL   OPTION XL1
NCP1     TERMINAL  DCB=NCP1DCB,         POINT TO PROPER 3705           X
                TERM=LNCP,IPLTXID=NCP1TXT
GRP1     GROUP MH=CICS,BUFSIZE=288,     POINT TO DEVICE MESSAGE HANDLERX
                OPACING=2               DEFINE HOST PACING
L1       TERMINAL  TERM=LINE,           DEFINE FIRST SDLC LINE         X
                GROUP=GRP1,             POINT TO PROPER GROUP          X
                RLN=1,                  FIRST LINE                     X
                ACTIVE=YES              ACTIVATE LINE AUTOMATICALLY
PU1      TERMINAL  TERM=PUNT            3790 PHYSICAL UNIT -
P1T1     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(80)
P1T2     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(80)
P1T3     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(80)
P1T4     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(00)
         SPACE 2
PU2      TERMINAL  TERM=PUNT            SECOND 3790 ON THE SAME LINE
P2T1     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(80)
P2T2     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,         X
                TCMSESN=LUINIT,OPDATA=(00)
L2       TERMINAL  TERM=LINE,GROUP=GRP1,RLN=2
PU3      TERMINAL  TERM=PUNT            FIRST 3767
P3T1     TERMINAL  TERM=LUNT,GROUP=GRP1,QBY=T,QUEUES=MR,RLN=2,         X
                TCMSESN=LUINIT,OPDATA=(00)
PU4      TERMINAL  TERM=PUNT,ACTIVE=YES     FIRST 3770
P4T1     TERMINAL  TERM=LUNT,GROUP=GRP1,QBY=T,QUEUES=MR,RLN=2,         X
                TCMSESN=LUINIT,OPDATA=(00)
CLNE     TPROCESS  PCB=CICSPCB,         POINT TO PROCESS CONTROL BLOCK *
                ALTDEST=CLNE,           REROUTE BACK AT QUEUE CLEAN-UP *
                QUEUES=MR
TLNE     TPROCESS  PCB=CICSPCB
SSCP     TERMINAL  TERM=SSCP
         EJECT
***********************************************************************
*
*        SYSTEM SERVICES CONTROL POINT MESSAGE HANDLER
*
***********************************************************************
         IEDMHGEN  SSCP=YES,TOTE=NO     INVOKE IBM-SUPPLIED SSCP MH
         EJECT
```

```
**********************************************************************
*
*        DEVICE MESSAGE HANDLER
*
**********************************************************************
CICS     STARTMH LC=OUT,DFC=FULL,LU=YES
         SPACE 3
**********************************************************************
*
*        INPUT MESSAGE HANDLER
*
**********************************************************************
         INHDR
         SETSCAN 0                      RETURN ADDRESS OF DATA
         LTR   RRETURN,RRETURN          ZERO LENGTH BUFFER
         BM    INMSG                    BRANCH IF YES
         LA    RSCAN,ONE(RRETURN)       SET SCAN POINTER
         IEDRH RHIND=(+DFC)             GET RH
         LTR   RRETURN,RRETURN          DFC COMMAND
         BNZ   NOTDFC                   BRANCH IF NO
         IEDRH RHIND=(+EXR)             GET RH
         LTR   RRETURN,RRETURN          EXCEPTION REQUEST
         BNZ   NOTEXR1                  BRANCH IF NO
         SETSCAN 4                      POINT TO COMMAND BYTE
         LA    RSCAN,FOUR(RSCAN)        UPDATE SCAN POINTER
NOTEXR1  EQU   *
         MSGTYPE X'C9'                  SIGNAL COMMAND
         IEDRELS                        START OUTPUT
         IEDRH BSTATE=YES               GET THE BRACKET STATE
         N     RRETURN,RTRMASK          TURN OFF RTR STATE
         CLM   RRETURN,4,PBETB          PENDING BETB
         BE    NOHOLD                   BRANCH IF YES
         CLM   RRETURN,4,BETB           BETB
         BE    NOHOLD                   BRANCH IF YES
         LOCOPT SNACTL                  GET OPTION FIELD
         TM    ZERO(RRETURN),SNASCD     CD ALREADY SENT
         BO    NOHOLD                   BRANCH IF YES
         OI    ZERO(RRETURN),SNASCD     SET CD SENT
         TERRSET                        SET USER ERROR BIT
         HOLD                           PREVENT OUTPUT
NOHOLD   EQU   *
         MSGTYPE X'04'                  LUSTAT COMMAND
         IEDRELS                        START OUTPUT
         CLC   ONE(TWO,RSCAN),SENS1     COMPONENT AVAILABLE
         BE    NOHALT                   BRANCH IF YES
         CLC   ONE(TWO,RSCAN),SENS2     NO DATA TO XMIT
         BE    NOHALT                   BRANCH IF YES
         LOCOPT SNACTL                  GET OPTION FIELD
         NI    ZERO(RRETURN),255-(SNASCD+SNASDS)
*                                       RESET STATES
         IEDHALT
```

```
NOHALT   EQU   *
         MSGTYPE X'C1'                SHUTDOWN COMPLETE COMMAND
         HOLD                         STOP OUTPUT
         MSGTYPE ,                    ALL OTHER DFC
         B     INBUF                  NO PROCESSING TO DO
         EJECT
NOTDFC   EQU   *
         IEDRH BSTATE=YES             GET BRACKET STATE
         N     RRETURN,RTRMASK        TURN OFF RTR STATE
         CLM   RRETURN,4,BETB         BETWEEN BRACKETS
         BE    NOTINB                 BRANCH IF YES
         CLM   RRETURN,4,PBETB        PENDING BETWEEN BRACKETS
         BE    NOTINB                 BRANCH IF YES
         LOCOPT SNACTL                GET OPTION FIELD
         OI    ZERO(RRETURN),SNASCD   SET CHANGE DIRECTION STATE
NOTINB   EQU   *
         IEDRH RHIND=(+EXR)           GET RH
         LTR   RRETURN,RRETURN        EXCEPTION REQUEST
         BZ    INMSG                  BRANCH IF YES
         CLI   ZERO(RSCAN),ZERO       NULL RU
         BE    INBUF                  BRANCH IF YES
         LR    RSCANSVE,RSCAN         SAVE THE SCAN POINTER
         MSGEDIT ((I,XL2'0000'))      INSERT NO FMH CCB
         LA    RSCAN,TWO(RSCAN)       POINT TO START OF FMH
         IEDRH RHIND=(+FMH)           GET RH
         LTR   RRETURN,RRETURN        FMH PRESENT
         BNZ   NOTFMH                 BRANCH IF NO
         TM    FMHTYPE(RSCAN),FMHTYP1 TYPE 1 FMH
         BZ    NOTBDS                 BRANCH IF NO
         TM    FMHSEL(RSCAN),FMHBDS   BEGIN OF DATASET
         BZ    NOTBDS                 BRANCH IF NO
         IC    RWORK,FMHLEN(RSCAN)    GET FMH LENGTH
         STC   RWORK,SCANSET+7        SET AMOUNT FOR SETSCAN
SCANSET  SETSCAN 1,BLANK=NO           SCAN PAST FMH
         MSGEDIT ((I,C'BTCH'))        EDIT IN BATCH TRANSACTION NAME
NOTBDS   EQU   *
         OI    ZERO(RSCANSVE),CCBFMH  INDICATE FMH PRESENT
NOTFMH   EQU   *
         FORWARD DEST=C'CLNE'         SEND MESSAGE TO CICS
INBUF    INBUF
         IEDRH RHIND=(+CHNGDIR,+EB)   GET RH
         CLM   RRETURN,1,RETCD8       CD OR EB PRESENT
         BE    NOTCD                  BRANCH IF NO
         LOCOPT SNACTL                GET OPTION FIELD
         NI    ZERO(RRETURN),255-SNASCD RESET CD SENT
         TM    ZERO(RRETURN),SNALUT2  LU TYPE 2
         BZ    JUSTREL                BRANCH IF NO
         TERRSET                      SEND UNLOCK MSGGEN
JUSTREL  EQU   *
         IEDRELS                      START OUTPUT
NOTCD    EQU   *
INMSG    INMSG PATH=(SNACTL,X'80')    LU TYPE 2 INMSG
         CANCELMG X'00060577FF'       CANCEL ON AN ERROR
         IEDHALT  X'00060577FF'       END THE SESSION
         MSGGEN   X'0000080000',LUT2MSG,RH=X'038040'
```

```
INMSG1   INMSG                          ALL OTHER LU INMSG
         CANCELMG X'00060577FF'         CANCEL ON AN ERROR
         IEDHALT  X'00060577FF'         END THE SESSION
         MSGGEN   X'0000080000',RH=X'038020'
         INEND
         EJECT
*********************************************************************
*
*        OUTPUT MH
*
*********************************************************************
         SPACE 3
         OUTHDR
         SETSCAN 0                      TEST FOR DATA IN BUFFER
         LTR   RRETURN,RRETURN          ZERO LENGTH BUFFER
         BP    NOTZERO                  BRANCH IF NO
         IEDSENSE AREA=(4)              GET THE SNA SENSE
         CLM   RWORK,8,TEMPERR          RECOVERABLE ERROR
         BE    OUTMSG                   BRANCH IF YES
         LOCOPT SNACTL                  GET OPTION FIELD
         NI    ZERO(RRETURN),255-(SNASDS+SNASCD)
*                                       RESET STATES
         B     OUTMSG                   BRANCH
NOTZERO  EQU   *
         LA    RSCAN,ONE(RRETURN)       SET SCAN REG
         TM    ZERO(RSCAN),CCBFMH       FMH IN DATA
         BZ    NOFMH                    BRANCH IF NO
         IEDRH RHIND=(+FMH)             SET FMH PRESENT
         LOCOPT SNACTL                  GET OPTION FIELD
         TM    FMHTYPE(RSCAN),FMHTYP1   TYPE 1 FMH
         BZ    NOFMH                    BRANCH IF NO
         TM    FMHSEL(RSCAN),FMHBDS     BEGIN OF DATASET
         BZ    NOTBDS1                  BRANCH IF NO
         OI    ZERO(RRETURN),SNASDS     SET IN DATA SET
NOTBDS1  EQU   *
         TM    FMHSEL(RSCAN),FMHEDS     END OF DATASET
         BZ    NOFMH                    BRANCH IF NO
         NI    ZERO(RRETURN),255-SNASDS TURN OFF IN DATASET STATE
NOFMH    EQU   *
         IEDRH BSTATE=YES               GET BRACKET STATE
         N     RRETURN,RTRMASK          TURN OFF RTR STATE
         CLM   RRETURN,4,BETB           BETWEEN BRACKETS
         BNE   CHKEB                    BRANCH IF NO
         IEDRH RHIND=(+BB)              SET BEGIN BRACKET
CHKEB    EQU   *
         LOCOPT SNACTL                  GET OPTION FIELD
         TM    ZERO(RRETURN),SNASDS     IN DATASET STATE
         BO    REMCCB                   BRANCH IF YES
         NI    ZERO(RRETURN),255-(SNASCD+SNASDS)
*                                       RESET STATES
         IEDRH RHIND=(+EB)              SET END OF BRACKET
REMCCB   EQU   *
         MSGEDIT ((R,,SCAN,(2)))        REMOVE CCB
         OUTBUF PATH=(SNACTL,X'01')     EXECUTE IF CD REQUIRED
         IEDRH RHIND=(*CHNGDIR)         INSERT CD IN LAST OF CHAIN
         L     RWORK,IEDADBUF           GET CURRENT BUFFER
         TM    PRFSTAT1(RWORK),PRFNLSTN LAST BUFFER IN MESSAGE
         BO    OUTMSG                   BRANCH IF NO
         LOCOPT SNACTL                  GET OPTION FIELD
         NI    ZERO(RRETURN),255-SNARCD RESET OPTION SWITCH
         OI    ZERO(RRETURN),SNASCD     SET CD SENT
OUTMSG   OUTMSG
         HOLD  X'0004000002',RELEASE    TEMP ERROR WAIT FOR LUSTAT
         HOLD  X'0004000012',RELEASE    TEMP ERROR WAIT FOR LUSTAT
         HOLD  X'0004000013',RELEASE    BRACKET CONTENTION WAIT FOR EB
         HOLD  X'0000006000',INTVL=10   RETRY AFTER WAIT
         IEDHALT X'0000010600'          END THE SESSION ON NON
*                                       RECOVERABLE ERRORS
         MSGGEN X'0000040008',MSG2,RH=X'0B8040'
*                                       ABORT THE DATASET ON ERROR
         MSGGEN X'0000040008',C'FMH ERROR DS ABORTED',
               RH=X'0380C0'
*                                       INFORM THE OPERATOR
         OUTEND
         EJECT
**********************************************************************
*
*        MESSAGE HANDLER FOR CICS APPLICATION PROGRAM
*
**********************************************************************
AMH      STARTMH
         INHDR
         FORWARD    DEST=PUT
         INEND
         OUTHDR
         OUTEND
         EJECT
LUT2MSG  DC    X'02F1C3'                RESET THE KEY BOARD
LUT1MSG  DC    X'0115'                  RETURN THE CARRIAGE
         DS    0F                       FORCE ALIGNMENT
RTRMASK  DC    X'FFEFFFFF'              MASK TO AND OFF RTR STATE
BETB     DC    X'00'                    COMPARE FOR BETB
PBETB    DC    X'20'                    COMPARE FOR PENDING BETB
SENS1    DC    X'0001'                  COMPONENT AVAILABLE
SENS2    DC    X'0002'                  NO DATA TO XMIT
TEMPERR  DC    X'08'                    REQUEST REJECT ERRORB
RETCD8   DC    X'08'
MSG1     DC    X'000000'                MSG AREA
MSG2     DC    X'0606010000A000'        ABORT DATASET FMH
         EJECT
         END
```

SAMPLE 2: DFHSPTM2 — SAMPLE TCAM MCP FOR TCAM DIRECT

```
***********************************************************************
*
*
*
*  NAME - CICS SAMPLE TCAM MESSAGE CONTROL PROGRAM AND MESSAGE HANDLERS
*         (CONTROLLED BY APPLICATION PROGRAMS)
*
*  PURPOSE - THE PURPOSE OF THIS SAMPLE IS TO DEMONSTRATE TO THE CICS
*        USER WHAT IS REQUIRED FOR THE CREATION OF A SIMPLE CICS SNA
*        NETWORK USING TCAM.
THE SAMPLE DEMONSTRATES HALF-DUPLEX
*        FLIP/FLOP MODE WITH BRACKETS.
*
*
*  FUNCTIONS - THE FUNCTIONS ARE AS FOLLOWS:
*
*     INTRODUCTION - DEFINE THE OVERALL SYSTEM PARAMETERS, INITIALIZE
*        THE SYSTEM, AND START MESSAGE TRAFFIC.
*
*     DEFINITION - DESCRIBE THE SPECIFICS OF THE NETWORK, THE MESSAGE
*        QUEUES, AND THE NECESSARY CONTROL BLOCKS.
*
*     DEVICE MESSAGE HANDLER - INSERT THE COMMUNICATIONS CONTROL BYTES
*        AND ROUTE THE MESSAGE FOR INPUT,  REMOVE THE CCB AND ROUTE
*        THE MESSAGE TO ITS PROPER DESTINATION ON OUTPUT.
THE DMH IS
*        NAMED 'CICS' SO THAT AN LU CAN LOGON TO 'CICS'.
*        THIS MH SUPPORTS LU TYPE0, TYPE1, AND TYPE2. LU TYPE2 IS THE
*        3270 DATA STREAM EMULATOR. THE SUPPORT IS DESIGNED TO ALLOW
*        THE TRANSACTION TO CONTROL THE LU.
OTHER MODES OF OPERATION
*        ARE POSSIBLE BY USING DIFFERENT MH OPTIONS.
THE LU TYPE1
*        BATCH SUPPORT ASSUMES A SINGLE TRANSACTION WILL HANDLE THE
*        BATCH INPUT.
THE MH WILL EDIT THE TRANSACTION NAME INTO
*        THE FIRST CHAIN OF THE DATASET.
THEREFORE IT IS NOT NECESSARY
*        TO PLACE A TRANSACTION NAME INTO THE DATASET.
*        THIS MH ASSUMES THAT THE TERMINAL WILL BE LOGICALLY TIED
*        TO A TRANSACTION FOR THE DURATION OF A BRACKET.
ADDITIONAL
*        FLOW CONTROL WOULD HAVE TO BE ADDED TO HANDLE MESSAGE
*        SWITCHING OR HOST INITIATED BRACKETS.
*
*
*     APPLICATION MESSAGE HANDLER - ROUTE MESSAGES FROM THE INPUT QUEUE
*        TO CICS AND FROM CICS TO THE APPROPRIATE OUTPUT QUEUE.
*
*
*     THE SSCP MESSAGE HANDLER  - UTILIZES THE IBM-SUPPLIED MH  TO
*        PERFORM THE NECESSARY ROUTING AND ANALYSIS FUNCTIONS.
```

```
COMPLEX
*       USER SYSTEMS MAY REQUIRE THIS TO BE MODIFIED BY THE USER.

*
*
* NOTES -
*
*   . CONVENTIONS -
*
*       REGISTER 2 IS USED AS THE DCB REGISTER
*
*       REGISTER 3 USED AS INTERNAL LINKAGE REGISTER
*
*       REGISTER 4 USED AS INTERNAL WORK REGISTER
*
*       REGISTER 5 USED AS INTERNAL WORK REGISTER
*
*       REGISTER 6 USED AS THE SCAN REGISTER
*
*    DEFAULTS -
*
*       MACRO DEFAULTS ARE USED WHEREVER REASONABLE
*
*
*    EXITS -
*
*       NORMAL -
*
*       RETURN TO THE MVS SUPERVISOR WHEN SHUTDOWN IS COMPLETE
*
*    ERROR -
*
*       X'FFF' - ABEND ON INTRO FAILURE
*
*       X'FFE' - ABEND ON MESSAGE QUEUE DCB OPEN FAILURE
*
*       X'FFD' - ABEND ON 3705 DCB OPEN FAILURE
*
*
*
*
***********************************************************************
CICSTCAM CSECT
RDCB     EQU   2                        DCB REGISTER
LINKREG  EQU   3                        INTERNAL LINKAGE REGISTER
RWORK    EQU   4                        INTERNAL WORK REGISTER
RSCANSVE EQU   5                        SAVED SCAN POINTER REGISTER
RSCAN    EQU   6                        SCAN POINTER REGISTER
RRETURN  EQU   15
OPEN     EQU   X'10'                    DCB OPEN FLAG
DCBOFLGS EQU   X'30'                    OPEN FLAGS OFFSETF
WORD     EQU   4                        OFFSET
         SPACE 2
*        CCB BYTE 0
```

```
         SPACE
CCBFMH   EQU   X'01'                   FORMATTED HEADER
CCBDISC  EQU   X'08'                   DISCONNECT
         SPACE 2
*        CCB BYTE 1
CCBEB    EQU  X'01'                    WRITE LAST SPECIFIED
CCBCD    EQU  X'02'                    WRITE WITH READ SPECIFIED
         SPACE 2
FMHLEN   EQU   0                       BYTE 0 OF FMH
FMHTYPE  EQU   1                       BYTE 1 OF FMH
FMHSEL   EQU   2                       BYTE 2 OF FMH
FMHSTCK  EQU   3                       BYTE 3 OF FMH
FMHPROP  EQU   4                       BYTE 4 OF FMH
FMHTYP1  EQU   X'01'                   TYPE 1 FMH
FMHBDS   EQU   X'40'                   BEGIN DATASET FMH
PRFSTAT1 EQU   X'14'                   STATUS BYTE OFFSET
PRFNLSTN EQU   X'02'                   NOT LAST INDICATOR
ZERO     EQU   0
ONE      EQU   1
TWO      EQU   2
FOUR     EQU   4
*********************************************************************
*        SNACTL OPTION FIELD USAGE
SNARCD   EQU   X'01'                   REMEMBER TO SET CD
SNASCD   EQU   X'02'                   CD HAS BEEN SENT, TCAM CAN'T
*                                      SEND ANY DATA
SNALUT2  EQU   X'80'                   TYPE 2 LU A 3270 DSE
*********************************************************************
         EJECT
         INTRO PROGID=CICSTCAM,                                        X
               UNITSZ=160,             BUFFER UNIT SIZE                X
               LNUNITS=100,            SEE TCAM SYSTEM PGMER'S GUIDE   X
               BRACKET=YES,            INCLUDE BRACKET STATE MANAGER   X
               BTRACE=500,             PIU TRACE ENTRIES               X
               CIB=5,                  MAX 5 ACTIVE OPERATOR COMMANDS  X
               COMWRTE=YES,            INCLUDE SERVICE AID WRITER      X
               CONTROL=OPCTL,          ALLOW COMMANDS FROM OTHERS      X
               CPB=20,                 ACTIVE DISK CHANNEL PROGRAMS    X
               DISK=YES,               DISK MESSAGE QUEUES             X
               DLQ=0,                  NO DEAD LETTER QUEUE            X
               DTRACE=500,             DISPATCHER TRACE ENTRIES        X
               FEATURE=(NODIAL,NO2741,,,ONLY3705,ONLYSNA),             X
               MSUNITS=100,            BUFFER UNITS FOR CORE QUEUE     X
               MAXSUBA=3,              OBTAIN FROM NCP GENERATION      X
               PLCBNO=20,              PSEUDO LCBS NEEDED FOR I/O      X
               PRIMARY=SYSCON,         OPERATOR CONTROL CONSOLE        X
               SIBCNT=25,              MAXIMUM SNA SESSIONS PERMITTED  X
               SUBAREA=1               OBTAIN FROM NCP GENERATION
         LTR   RRETURN,RRETURN         WAS INTRO SUCCESSFUL
         BZ    OKINTRO                 IF SO, CARRY ON
         STH   RRETURN,DEBUG           MAKE RETURN CODE VISIBLE
         ABEND 4095,DUMP               OTHERWISE, PUNT WITH X'FFF'
OKINTRO  OPEN  (MSGQUEUE,(INOUT))      OPEN MESSAGE QUEUES DATA SET
         LA    RDCB,MSGQUEUE           POINT TO DCB
         TM    DCBOFLGS(RDCB),OPEN     WAS OPEN SUCCESSFUL
         BO    OKOPENQS                IF SO, CARRY ON
         ABEND 4094,DUMP              OTHERWISE, PUNT WITH X'FFE'
OKOPENQS OPEN  (NCP1DCB,(INOUT))      OPEN 3705 FOR COMMUNICATIONS
         LA    RDCB,NCP1DCB           POINT TO 3705 DCB
         TM    DCBOFLGS(RDCB),OPEN    WAS OPEN SUCCESSFUL
         BO    OKOPEN05               IF SO, CARRY ON
         ABEND 4093,DUMP              OTHERWISE, PUNT WITH X'FFD'
OKOPEN05 READY                        LET TCAM START TRAFFIC
         CLOSE (NCP1DCB,,MSGQUEUE,)
         RETURN     (14,12)           RELINQUISH CONTROL
         EJECT
***********************************************************************
*
*        DEFINE THE CONFIGURATION OF THE NETWORK - PHYSICAL AND LOGICAL
*
***********************************************************************
         SPACE 2
         DS    0D
         DC    C'  RETURN CODE = '    TRIGGER FOR DUMP SCANNING
DEBUG    DS    H                      TO CONTAIN INTRO RETURN CODE
         DC    C'  '                  SPACING AROUND MESSAGE
MSGQUEUE DCB   DSORG=TQ,              MESSAGE QUEUE DATA SET         X
               DDNAME=MSGQUEUE,                                      X
               MACRF=(G,P),                                          X
               OPTCD=R                REUSABLE DISK QUEUES
NCP1DCB  DCB   DSORG=TR,              3705 COMMUNICATIONS CONTROLLER X
               DDNAME=DDNCP1,                                        X
               MACRF=(G,P)
CICSPCB  PCB   MH=AMH,BUFSIZE=2000    APPLICATION PROGRAM MH
         TTABLE LAST=SSCP             TERMINAL TABLE START AND END
SNACTL   OPTION XL1
NCP1     TERMINAL  DCB=NCP1DCB,       POINT TO PROPER 3705           X
               TERM=LNCP,IPLTXID=NCP1TXT
GRP1     GROUP MH=CICS,BUFSIZE=288,   POINT TO DEVICE MESSAGE HANDLERX
               OPACING=2              DEFINE HOST PACING
L1       TERMINAL  TERM=LINE,         DEFINE FIRST SDLC LINE         X
               GROUP=GRP1,            POINT TO PROPER GROUP          X
               RLN=1,                 FIRST LINE                     X
               ACTIVE=YES             ACTIVATE LINE AUTOMATICALLY
PU1      TERMINAL  TERM=PUNT          3790 PHYSICAL UNIT -
P1T1     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(80)
P1T2     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(80)
P1T3     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(80)
P1T4     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(00)
         SPACE 2
PU2      TERMINAL  TERM=PUNT          SECOND 3790 ON THE SAME LINE
P2T1     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(80)
P2T2     TERMINAL  GROUP=GRP1,QBY=T,QUEUES=MR,RLN=1,TERM=LUNT,       X
               TCMSESN=LUINIT,OPDATA=(00)
L2       TERMINAL  TERM=LINE,GROUP=GRP1,RLN=2
PU3      TERMINAL  TERM=PUNT          FIRST 3767
P3T1     TERMINAL  TERM=LUNT,GROUP=GRP1,QBY=T,QUEUES=MR,RLN=2,       X
```

```
                TCMSESN=LUINIT,OPDATA=(00)
PU4      TERMINAL   TERM=PUNT,ACTIVE=YES      FIRST 3770
P4T1     TERMINAL   TERM=LUNT,GROUP=GRP1,QBY=T,QUEUES=MR,RLN=2,        X
                TCMSESN=LUINIT,OPDATA=(00)
CLNE     TPROCESS   PCB=CICSPCB,           POINT TO PROCESS CONTROL BLOCK *
                ALTDEST=CLNE,              REROUTE BACK AT QUEUE CLEAN-UP *
                QUEUES=MR
TLNE     TPROCESS   PCB=CICSPCB
SSCP     TERMINAL   TERM=SSCP
         EJECT
***********************************************************************
*
*        SYSTEM SERVICES CONTROL POINT MESSAGE HANDLER
*
***********************************************************************
         IEDMHGEN   SSCP=YES,TOTE=NO       INVOKE IBM-SUPPLIED SSCP MH
         EJECT
***********************************************************************
*
*        DEVICE MESSAGE HANDLER
*
***********************************************************************
CICS     STARTMH LC=OUT,DFC=FULL,LU=YES
         SPACE 3
***********************************************************************
*
*        INPUT MESSAGE HANDLER
*
***********************************************************************
         INHDR

         SETSCAN 0                         RETURN ADDRESS OF DATA
         LTR   RRETURN,RRETURN             ZERO LENGTH BUFFER
         BM    INMSG                       BRANCH IF YES
         LA    RSCAN,ONE(RRETURN)          SET SCAN POINTER
         IEDRH RHIND=(+DFC)                GET RH
         LTR   RRETURN,RRETURN             DFC COMMAND
         BNZ   NOTDFC                      BRANCH IF NO
         IEDRH RHIND=(+EXR)                GET RH
         LTR   RRETURN,RRETURN             EXCEPTION REQUEST
         BNZ   NOTEXR1                     BRANCH IF NO
         SETSCAN 4                         POINT TO COMMAND BYTE
         LA    RSCAN,FOUR(RSCAN)           UPDATE SCAN POINTER
NOTEXR1  EQU   *
         MSGTYPE X'C9'                     SIGNAL COMMAND
         IEDRH BSTATE=YES                  GET THE BRACKET STATE
         N     RRETURN,RTRMASK             TURN OFF RTR STATE
         CLM   RRETURN,4,PBETB             PENDING BETB
         BE    NOHOLD                      BRANCH IF YES
         CLM   RRETURN,4,BETB              BETB
         BE    NOHOLD                      BRANCH IF YES
         LOCOPT SNACTL                     GET OPTION FIELD
         TM    ZERO(RRETURN),SNASCD        CD ALREADY SENT
         BO    NOHOLD                      BRANCH IF YES
         TM    ZERO(RRETURN),SNALUT2       TYPE 2 LU
         BZ    LUTYP1                      BRANCH IF NO
         MVC   MSG1(L'LUT2MSG),LUT2MSG  SET LU TYPE 2 MESSAGE
         B     SETERR                   BRANCH
LUTYP1   EQU   *
         MVC   MSG1(L'LUT1MSG),LUT1MSG  SET LU TYPE 1 MESSAGE
SETERR   EQU   *
         OI    ZERO(RRETURN),SNASCD     SET CD SENT
         TERRSET                        SET USER ERROR BIT
         HOLD                           PREVENT OUTPUT
NOHOLD   EQU   *
         MSGTYPE X'04'                  LUSTAT COMMAND
         IEDRELS                        START OUTPUT
         CLC   ONE(TWO,RSCAN),SENS1     COMPONET AVAILABLE
         BE    NOHALT                   BRANCH IF YES
         CLC   ONE(TWO,RSCAN),SENS2     NO DATA TO XMIT
         BE    NOHALT                   BRANCH IF YES
         IEDHALT
NOHALT   EQU   *
         MSGTYPE X'C1'                  SHUTDOWN COMPLETE COMMAND
         HOLD                           STOP OUTPUT
         MSGTYPE ,                      ALL OTHER DFC
         B     INBUF                    NO PROCESSING TO DO
         EJECT
NOTDFC   EQU   *
         IEDRH RHIND=(+EXR)             GET RH
         LTR   RRETURN,RRETURN          EXCEPTION REQUEST
         BZ    INMSG                    BRANCH IF YES
         CLI   ZERO(RSCAN),ZERO         NULL RU
         BE    INBUF                    BRANCH IF YES
         LR    RSCANSVE,RSCAN           SAVE THE SCAN POINTER
         MSGEDIT ((I,XL2'0000'))        INSERT NO FMH CCB
         LA    RSCAN,TWO(RSCAN)         POINT TO START OF FMH
         IEDRH RHIND=(+FMH)             GET RH
         LTR   RRETURN,RRETURN          FMH PRESENT
         BNZ   NOTFMH                   BRANCH IF NO
         TM    FMHTYPE(RSCAN),FMHTYP1   TYPE 1 FMH
         BZ    NOTBDS                   BRANCH IF NO
         TM    FMHSEL(RSCAN),FMHBDS     BEGIN OF DATASET
         BZ    NOTBDS                   BRANCH IF NO
         IC    RWORK,FMHLEN(RSCAN)      GET FMH LENGTH
         STC   RWORK,SCANSET+7          SET AMOUNT FOR SETSCAN
SCANSET  SETSCAN 1,BLANK=NO             SCAN PAST FMH
         MSGEDIT ((I,C'BTCH'))          EDIT IN BATCH TRANSACTION NAME
NOTBDS   EQU   *
         OI    ZERO(RSCANSVE),CCBFMH    INDICATE FMH PRESENT
NOTFMH   EQU   *
         FORWARD DEST=C'CLNE'           SEND MESSAGE TO CICS
INBUF    INBUF
         IEDRH RHIND=(+CHNGDIR)         GET RH
         LTR   RRETURN,RRETURN          CD PRESENT
         BNZ   NOTCD                    BRANCH IF NO
         LOCOPT SNACTL                  GET OPTION FIELD
         NI    ZERO(RRETURN),255-SNASCD RESET CD SENT
         IEDRELS                        START OUTPUT
NOTCD    EQU   *
```

```
INMSG    INMSG
         CANCELMG X'00060577FF'        CANCEL ON AN ERROR
         IEDHALT  X'00060577FF'        END THE SESSION
         MSGGEN   X'0000080000',MSG1,RH=X'038020'
         INEND
         EJECT
*********************************************************************
*
*        OUTPUT MH
*
*********************************************************************
         SPACE 3
         OUTHDR
         SETSCAN 0                     TEST FOR DATA IN BUFFER
         LTR   RRETURN,RRETURN         ZERO LENGTH BUFFER
         BM    OUTMSG                  BRANCH IF YES
         LA    RSCAN,ONE(RRETURN)      SET SCAN REG
         TM    ZERO(RSCAN),CCBDISC     SESSION END REQUESTED
         BZ    NOTDISC                 BRANCH IF NO
         CANCELMG
         B     OUTMSG                  STOP THE MESSAGE, END THE
*                                      SESSION AND QUIT PROCESSING
NOTDISC  EQU   *
         TM    ZERO(RSCAN),CCBFMH      FMH IN DATA
         BZ    NOFMH                   BRANCH IF NO
         IEDRH RHIND=(+FMH)            SET FMH PRESENT
NOFMH    EQU   *
         IEDRH BSTATE=YES              GET BRACKET STATE
         N     RRETURN,RTRMASK         TURN OFF RTR STATE
         CLM   RRETURN,4,BETB          BETWEEN BRACKETS
         BNE   CHKEB                   BRANCH IF NO
         IEDRH RHIND=(+BB)             SET BEGIN BRACKET
CHKEB    EQU   *
         TM    ONE(RSCAN),CCBEB        END OF TRANSACTION
         BZ    CHKCD                   BRANCH IF NO
         IEDRH RHIND=(+EB)             SET END OF BRACKET
         LOCOPT SNACTL                 GET OPTION FIELD
         NI    ZERO(RRETURN),255-SNASCD RESET CD SENT
         B     REMCCB                  GO REMOVE CCB
CHKCD    EQU   *
         TM    ONE(RSCAN),CCBCD        INPUT FROM TERMINAL WANTED
         BZ    REMCCB                  BRANCH IF NO
         HOLD                          STOP FURTHER OUTPUT
         LOCOPT SNACTL                 GET OPTION FIELD
         OI    ZERO(RRETURN),SNARCD    SET PATH SWITCH TO SET CD
REMCCB   EQU   *
         MSGEDIT ((R,,SCAN,(2)))       REMOVE CCB
         OUTBUF PATH=(SNACTL,X'01')    EXECUTE IF CD REQUIRED
         IEDRH RHIND=(*CHNGDIR)        INSERT CD IN LAST OF CHAIN
         L     RWORK,IEDADBUF          GET CURRENT BUFFER
         TM    PRFSTAT1(RWORK),PRFNLSTN LAST BUFFER IN MESSAGE
         BO    OUTMSG                  BRANCH IF NO
         LOCOPT SNACTL                 GET OPTION FIELD
         NI    ZERO(RRETURN),255-SNARCD RESET OPTION SWITCH
         OI    ZERO(RRETURN),SNASCD    SET CD SENT
```

```
OUTMSG   OUTMSG
         HOLD  X'0004000002',RELEASE    TEMP ERROR WAIT FOR LUSTAT
         HOLD  X'0004000012',RELEASE    TEMP ERROR WAIT FOR LUSTAT
         HOLD  X'0000006000',INTVL=10   RETRY AFTER WAIT
         IEDHALT X'0000009000',CONNECT=AND
*                                       END THE SESSION IF REQUESTED
         IEDHALT X'0000050600'          END THE SESSION ON NON
*                                       RECOVERABLE ERRORS
         OUTEND
         EJECT
***********************************************************************
*
*        MESSAGE HANDLER FOR CICS APPLICATION PROGRAM
*
***********************************************************************
AMH      STARTMH
         INHDR
         FORWARD    DEST=PUT
         INEND
         OUTHDR
         OUTEND
         EJECT
LUT2MSG  DC    X'02F1C3'                RESET THE KEY BOARD
LUT1MSG  DC    X'0115'                  RETURN THE CARRIAGE
         DS    0F                       FORCE ALIGNMENT
RTRMASK  DC    X'FFEFFFFF'              MASK TO AND OFF RTR STATE
BETB     DC    X'00'                    COMPARE FOR BETB
PBETB    DC    X'20'                    COMPARE FOR PENDING BETB
SENS1    DC    X'0001'                  COMPONENT AVAILABLE
```

End of Diagnosis, Modification and Tuning Information

Appendix C. Macro instruction format

General-Use Programming Interface

The CICS macro instructions are written in assembler language in the following format:

Name	Operation	Operand	Comments
blank or symbol	DFHxxxxx	One or more operands separated by commas	

Use the operand field to specify the services and options to be generated. Operands are always in a keyword format. Specify any parameters according to the following general rules:

- If the parameter associated with the operand is written in all capital letters (for example, TYPE=INITIAL), specify the operand and parameter exactly as shown.
- If the parameter associated with the operand is written in lowercase letters, specify the operand exactly as shown and substitute the indicated value, address, or name for the lowercase letters (for example, FILE=name).
- Code commas and parentheses exactly as shown, but omit a comma following the last operand. The use of commas and parentheses is indicated by brackets and braces, exactly as for operands. You may omit the parentheses when only one parameter of a particular operand is used.
- Because a blank character indicates the end of the operand field, the operand field must not contain blanks except within quotes, after a comma on a continued line, or after the last operand of the macro instruction. Start the first operand on a continuation line in column 16.
- When you write a CICS macro instruction on more than one line, each line containing part of the macro instruction (except the last line) must contain a character (for example, an asterisk) in column 72, indicating that the macro instruction is continued on the next line.

Syntax notation

The symbols [ ], { }, | and ,... are used in this publication to help define the macro instructions. **Do not code these symbols**; they act only to indicate how a macro instruction can be written; their definitions are given below:

[ ]	indicates optional operands. You may code the operand enclosed in the brackets (for example, [FB]) or not, depending on whether you require the associated option. If more than one item is enclosed within brackets (for example, [BLOCKED|UNBLOCKED]), you can code one or none of the items. Any default value available is indicated by an underscore and will be taken if you do not code an option from the group.
{ }	indicates that you must make a choice. You can code one of the operands from the list within braces separated by a | symbol (for example, {YES|NO}), depending on which of the associated services you require. Any default value is indicated by an underscore.
|	indicates that you must choose between the operands that are separated by this symbol.
,...	indicates that you can code more than one set of operands in the same macro instruction.

To simplify the syntax notation in the case where you can code one or more operands, the notation:

```
PARM=([A][,B][,C][,D])
```

indicates that you can code any number or none of A,B,C, or D. Do not code any leading comma. If you code only one operand, you need not code the enclosing parentheses.

For example:

```
PARM=A
PARM=(A,B)
PARM=(B,D)
PARM=(C)
```

are all valid interpretations of the above notation.

The lowercase character "b" is used in some places to indicate a blank character.

End of General-Use Programming Interface

Appendix D. Coding entries in the VTAM LOGON mode table

This appendix shows you what you must have coded in your VTAM logmode table for a terminal for which you want to use automatic installation.

CICS uses the logmode data when processing an automatic installation request, and automatic installation will only function properly if the information is correct. CICS conforms to VTAM standards.

Two sections at the end of this Appendix show examples of matching entries.

The following tables show, for a variety of possible terminal models, what you must have coded in the MODEENT macros that define your logmode table if you want to use automatic installation. Between them they show the values that must be specified for each of the operands of the MODEENT macro. Where all bit settings of an operand's value have significance for CICS, the data is shown in hexadecimal form. If some of an operand's bit settings are not significant to CICS, its data bytes are shown as bit patterns. The bit settings that have significance for CICS are shown set to the values CICS expects. Those bits that have no significance to CICS are shown as periods. Thus, for example:

```
01..0011
```

shows that six bits in the subject byte must be given specific values; the remaining two have no significance.

Some of the examples shown here correspond exactly to entries in the IBM-supplied logon mode table called ISTINCLM. Where this is so, the table gives the name of the entry in ISTINCLM.

The PSERVIC setting shows fields called aaaaaaaa, bbbbbbbb, and so on. The contents of these vary according to certain specifications of attributes of LUTYPE2 and LUTYPE3 terminals. You can work out the values you need by looking at "PSERVIC values for LUTYPE0, LUTYPE2, and LUTYPE3 devices" on page 541.

Device definition options and pointers to related LOGMODE data

Search the list given in Figure 45 for the combination of options that represents the typeterm you want to use for an automatic installation model. When you find the right one, use the number to its right to locate, in Table 6 on page 537, what has to be coded in the VTAM MODEENT macros.

```
DEVICE(APPC) ........................................... 24
DEVICE(BCHLU) .......................................... 17
DEVICE(BCHLU) SESSIONTYPE(BATCHDI) ..................... 15
```

Figure 45 (Part 1 of 2). TYPETERMs with cross-references to VTAM mode entry specifications

```
DEVICE(BCHLU) SESSIONTYPE(USERPROG) ...................... 16
DEVICE(CONTLU) ........................................... 10
DEVICE(INTLU) ............................................ 11
DEVICE(LUTYPE0) .......................................... 25
DEVICE(LUTYPE2) .......................................... 18
DEVICE(LUTYPE3) .......................................... 19
DEVICE(LUTYPE4) .......................................... 12
DEVICE(SCSPRINT) .......................................11,13
DEVICE(TLX) .............................................. 8
DEVICE(TLX) SESSIONTYPE(CONTLU) .......................... 8
DEVICE(TLX) SESSIONTYPE(INTLU) ........................... 9
DEVICE(TWX) .............................................. 8
DEVICE(TWX) SESSIONTYPE(CONTLU) .......................... 8
DEVICE(TWX) SESSIONTYPE(INTLU) ........................... 9
DEVICE(3270) ............................................. 2
DEVICE(3270) BRACKET(NO) ................................. 1
DEVICE(3270P) ............................................ 2
DEVICE(3270P) BRACKET(NO) ................................ 1
DEVICE(3275) ............................................. 2
DEVICE(3275) BRACKET(NO) ................................. 1
DEVICE(3600) ............................................. 22
DEVICE(3600) ............................................. 23
DEVICE(3600) SESSIONTYPE(PIPELINE) ....................... 21
DEVICE(3600) SESSIONTYPE(PIPELN) ......................... 21
DEVICE(3614) ............................................. 3
DEVICE(3650) SESSIONTYPE(PIPELINE) ....................... 21
DEVICE(3650) SESSIONTYPE(PIPELN) ......................... 21
DEVICE(3650) SESSIONTYPE(USERPROG) BRACKET(YES) .......... 6
DEVICE(3650) SESSIONTYPE(USERPROG) BRACKET(NO) ........... 7
DEVICE(3650) SESSIONTYPE(3270) ........................... 5
DEVICE(3650) SESSIONTYPE(3270) BRACKET(NO) ............... 4
DEVICE(3650) SESSIONTYPE(3653) ........................... 5
DEVICE(3650) SESSIONTYPE(3653) BRACKET(NO) ............... 4
DEVICE(3767) ............................................. 11
DEVICE(3767C) ............................................ 10
DEVICE(3767I) ............................................ 11
DEVICE(3770) ............................................. 17
DEVICE(3770) SESSIONTYPE(BATCHDI) ........................ 15
DEVICE(3770) SESSIONTYPE(USERPROG) ....................... 16
DEVICE(3770B) ............................................ 17
DEVICE(3770B) SESSIONTYPE(BATCHDI) ....................... 15
DEVICE(3770B) SESSIONTYPE(USERPROG) ...................... 16
DEVICE(3770C) ............................................ 10
DEVICE(3770I) ............................................ 11
DEVICE(3790) ............................................. 20
DEVICE(3790) SESSIONTYPE(BATCHDI) ........................ 14
DEVICE(3790) SESSIONTYPE(SCSPRT) ......................... 13
DEVICE(3790) SESSIONTYPE(SCSPRINT) ....................... 13
DEVICE(3790) SESSIONTYPE(USERPROG) ....................... 16
DEVICE(3790) SESSIONTYPE(3277CM) ......................... 18
DEVICE(3790) SESSIONTYPE(3284CM) ......................... 19
DEVICE(3790) SESSIONTYPE(3286CM) ......................... 19
```

Figure 45 (Part 2 of 2). TYPETERMs with cross-references to VTAM mode entry specifications

VTAM MODEENT macro operands

Table 6 shows the nature of the logmode table entry for each TYPETERM you might define. You should have reached this table by looking up the TYPETERM attributes in Figure 45 on page 535.

Look down the left side of the table for the reference number (RN) that brought you here from Figure 45 on page 535. When you find it, look across to the middle column. This shows the macro operands that affect the way CICS handles automatic installation. Your MODEENT macro entries for devices to be installed must match what is specified there. Any MODEENT macro entries not shown in the table, such as RUSIZES or PSERVIC for some reference numbers, are not tested by CICS. Any bit settings that do not matter to CICS during bind analysis for autoinstalled terminals appear as periods (.). In particular, under RN 18 and RN 19, PSERVIC byte 2 bit 0 is not checked during bind analysis. However, it should be set on if you require extended data stream support.

Note: Some fields in the PSERVIC data for LUTYPE2 and LUTYPE3 devices have values that depend on the ALTSCREEN and DEFSCREEN characteristics of the device. For this reason, you have to consult "PSERVIC values for LUTYPE0, LUTYPE2, and LUTYPE3 devices" on page 541 to find out the values you need to specify instead of aaaaaaaa, bbbbbbbb, cccccccc, dddddddd, and eeeeeeee.

The right hand column in the table names entries in the IBM-supplied logon mode table that could meet your needs. The IBM-supplied table is called ISTINCLM. For further VTAM information, see *ACF VTAM Planning and Installation Reference*, SC27-0584.

Table 6 (Page 1 of 5). Logon Mode Table and ISTINCLM Entries

RN	Logon mode table (MODEENT) macro entries that are needed for related TYPETERM definitions	Suitable supplied entries

1	FMPROF=X'02'
	TSPROF=X'02'
	PRIPROT=X'70'
	SECPROT=X'40'
	COMPROT=B'0000.000 00000.00'	

2	FMPROF=X'02'
	TSPROF=X'02'
	PRIPROT=X'71'
	SECPROT=X'40'
	COMPROT=B'0010.000 00000.00'
		DSILGMOD
		D4B32781
		D4B32782
		D4B32783
		D4B32784
		D4B32785
		NSX32702
		S3270

3	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B0'
	SECPROT=X'B0'
	COMPROT=B'0000.000 00000.00'	

Table 6 (Page 2 of 5). Logon Mode Table and ISTINCLM Entries

RN	Logon mode table (MODEENT) macro entries that are needed for related TYPETERM definitions	Suitable supplied entries

4	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B0'
	SECPROT=X'90'
	COMPROT=B'0100.000 00000.00'	

5	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0110.000 00000.00'	

6	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'31'
	SECPROT=X'30'
	COMPROT=B'0110.000 00000.00'
		INTRUSER

7	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'30'
	SECPROT=X'30'
	COMPROT=B'0100.000 00000.00'	

8	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0011.000 01000.00'
	PSERVIC=B'00000001 00000000 00000000 0000000.
	.... 00000000 00000000 00000000
	00000000 .... 00000000 00000000'	

9	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0011.000 10000.00'
	PSERVIC=B'00000001 00000000 00000000 0000000.
	.... 00000000 00000000 00000000
	00000000 .... 00000000 00000000'
		SCS

10	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0011.000 01000.00'
	PSERVIC=X'01'	

11	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0011.000 10000.00'
	PSERVIC=X'01'
		See note 2

Table 6 (Page 3 of 5). Logon Mode Table and ISTINCLM Entries

RN	Logon mode table (MODEENT) macro entries that are needed for related TYPETERM definitions	Suitable supplied entries

12	FMPROF=X'07'
	TSPROF=X'07'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0101.000 10000.01'
	PSERVIC=B'00000100 10101000 01000000 10100000
	.... 10101000 01000000 10100000
	00000000 .... 00001100 00000000'	

13	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0011.000 10000.00'
	PSERVIC=X'01'
		SCS3790
		See note 2

14	FMPROF=X'03'
	TSPROF=X'04'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0111.000 10000.00'
	PSERVIC=B'00000001 00110001 00011000 0100000.
	.... 00000000 10010010 00000000
	00000000 .... 00000000 01010000'	

15	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0111.000 10000.00'
	PSERVIC=B'00000001 00110001 00001100 0111000.
	.... 00000000 11010010 00000000
	00000000 .... 00000000 11010000'	

16	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0111.000 10000.00'	

17	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=X'90'
	COMPROT=B'0111.000 10000.00'
	PSERVIC=B'00000001 00100000 00000000 0000000.
	.... 00000000 11000010 00000000
	00000000 .... 00000000 11000000'	

Table 6 (Page 4 of 5). Logon Mode Table and ISTINCLM Entries

RN	Logon mode table (MODEENT) macro entries that are needed for related TYPETERM definitions	Suitable supplied entries

18	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=B'10..0000'
	COMPROT=B'0011.000 10000.00'
	PSERVIC=B'00000010 .0000000 00000000 00000000
	00000000 00000000 aaaaaaaa bbbbbbbb
	cccccccc dddddddd eeeeeeee'
		D329001
		D4A32771
		D4A32772
		D4A32781
		D4A32782
		D4A32783
		D4A32784
		D4A32785
		D4C32771
		D4C32772
		D4C32781
		D4C32782
		D4C32783
		D4C32784
		D4C32785
		D6327801
		D6327802
		D6327803
		D6327804
		D6327805
		EMUDPCX
		EMU3790
		SNX32701
		SNX32702
		SNX32703
		SNX32704
		SNX32705
		See note 1

19	FMPROF=X'03'
	TSPROF=X'03'
	PRIPROT=X'B1'
	SECPROT=B'10..0000'
	COMPROT=B'0011.000 10000.00'
	PSERVIC=B'00000011 .0000000 00000000 00000000
	00000000 00000000 aaaaaaaa bbbbbbbb
	cccccccc dddddddd eeeeeeee'
		BLK3790
		DSC2K
		DSC4K
		D6328902
		D6328904
		See note 1

20	FMPROF=X'04'
	TSPROF=X'03'
	PRIPROT=X'31'
	SECPROT=X'B0'
	COMPROT=B'0111.000'	

21	FMPROF=X'04'
	TSPROF=X'03'
	PRIPROT=X'50'
	SECPROT=X'10'
	COMPROT=B'0000.000 00000.00'	

22	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B0'
	SECPROT=X'B0'
	COMPROT=B'0100.000 00000.00'
		IBMS3650

Table 6 (Page 5 of 5). Logon Mode Table and ISTINCLM Entries		
RN	**Logon mode table (MODEENT) macro entries that are needed for related TYPETERM definitions**	**Suitable supplied entries**

23	FMPROF=X'04'
	TSPROF=X'04'
	PRIPROT=X'B1'
	SECPROT=X'B0'
	COMPROT=B'0111.000 00000.00'	

24	TYPE=X'00'
	FMPROF=X'13'
	TSPROF=X'07'
	PRIPROT=X'B0'
	SECPROT=X'B0'
	COMPROT=B'0101.000 10110.01'
	PSERVIC=X'060200000000000000002C00'	

25	FMPROF=X'02'
	TSPROF=X'02'
	PRIPROT=X'71'
	SECPROT=X'40'
	COMPROT=B'0010.000 00000.00'
	PSERVIC=B'00000000 .0000000 00000000 00000000
	00000000 00000000 aaaaaaaa bbbbbbbb
	cccccccc dddddddd eeeeeeee'
		D4B32782
		D4B32783
		D4B32784
		D4B32785
		NSX32702
		NSX32703
		NSX32704
		NSX32705

Notes:

1. PSERVIC (RN 18 and 19): BYTE 2 BIT 0 is not checked for bind analysis. However, this bit should be set on where extended data stream (EXTDS) support is required.

2. RN 11 or 13 is used to determine the MODEENT macro operands for device SCSPRINT. However, if you have specified any of the attributes EXTENDEDDS, COLOR, PROGSYMBOLS, HILIGHT, SOSI, OUTLINE, QUERY(COLD), or QUERY(ALL) for the TYPETERM, then the COMPROT parameter of RN 11|13 should be modified to read COMPROT=B'0111.000 10000.00'.

PSERVIC values for LUTYPE0, LUTYPE2, and LUTYPE3 devices

Table 7 shows you what to specify in the PSERVIC operand of the MODEENT macro for LUTYPE2, LUTYPE3, and local non-SNA 3270 devices.

The PSERVIC value should be compatible with what you specify as DEFSCREEN and ALTSCREEN on your autoinstall model's TYPETERM definition. Where no value is specified, CICS does not test the value. If you don't specify anything for a byte, CICS looks for 00000001 if the device is a Model 1, and 00000010 if the device is a Model 2. For comparison purposes, note that, in Table 6 on page 537, PSERVIC bytes 20-24 are represented as aaaaaaaa, bbbbbbbb, cccccccc, dddddddd, and eeeeeeee respectively. For a full definition of PSERVIC values, see *SNA — Sessions between Logical Units*, GC20-1868.

Table 7. PSERVIC summary for LUTYPE2, LUTYPE3, and local non-SNA 3270 devices				
Device model	**DEFSCREEN**	**ALTSCREEN**	**QUERY**	**PSERVIC Bind bytes 20-24**
0,2,3	00,00	?	?	Invalid
0,2,3	12,40	Blanks	?	0000000001
0,2,3	12,40	00,00	?	OC2800007E
0,2,3	12,40	YY,YY	?	OC28YYYY7F
0,2,3	12,40	Blanks	NO	0000000002
3,4,5	24,80	Blanks	COLD|ALL	0000000002
0,2	24,80	Blanks	COLD|ALL	0000000003
0,2,3,4,5	24,80	00,00	?	185000007E
0,2,3,4,5	24,80	YY,YY	?	1850YYYY7F
0,2,3,4,5	XX,XX	Blanks	?	XXXX00007E
0,2,3,4,5	XX,XX	00,00	?	XXXX00007E
0,2,3,4,5	XX,XX	YY,YY	?	XXXXYYYY7F

Notes:

1. 0 indicates a local non-SNA 3270 device, 2 indicates an LUTYPE2 device, and 3 indicates an LUTYPE3 device.
2. XX,XX indicates a screen size that is not 12,40 or 24,80.
3. YY,YY indicates a screen size that is not 00,00 or blanks.
4. ? indicates that the entry for ALTSCREEN may be any size, and the entry for QUERY may be ALL, COLD, or NO.

Table 8 shows the bind settings of autoinstall models against the CEDA parameters used to create them.

Table 8. PSERVIC summary for LUTYPE0, LUTYPE2, and LUTYPE3				
LUTYPE	**DEFSCREEN**	**ALTSCREEN**	**QUERY**	**PSERVIC Bind bytes 20-24**
0,2,3	00,00	00,00	?	0000000000
0,2,3	12,40	00,00	NO	0000000001
0,2,3	24,80	00,00	NO	0000000002, or 185000007E
3	24,80	00,00	COLD|ALL	0000000002
0,2	24,80	00,00	COLD|ALL	0000000003, or 185000007E (see note 4)
0,2,3	MM,MM	00,00	?	MMMM00007E
0,2,3	NN,NN	PP,PP	?	NNNNPPPP7F

Notes:

1. MM,MM is any pair of values other than 00,00 12,40 or 24,80
2. NN,NN or PP,PP is any pair of values other than 00,00
3. ? means that any QUERY value is allowed. However, if QUERY=COLD or QUERY=ALL is coded and a query is sent to a device that does not support it, the result is unpredictable.
4. The value 185000007E applies only if the query bit is on in PSERVIC byte 15.

Matching models and LOGMODE entries

This section contains a set of VTAM LOGMODE definitions, and their matching CICS Autoinstall model definitions. Each entry consists of a VTAM LOGMODE definition, the matching CICS TYPETERM/TERMINAL autoinstall model definition, and (for information) the BIND which CICS will send based on the specified model definition.

Note that the CICS-specific attributes are purely arbitrary. Only device attributes affect the matching algorithm. It is the responsibility of the autoinstall user program to distinguish between matching models.

```
******************************************************************
1) LOCAL NON-SNA 3277 / 3278 / 3279 (without special features)
******************************************************************

MT32772  MODEENT LOGMODE=MT32772,  3277/8      MODEL 2
               TYPE=1,
               FMPROF=X'02',
               TSPROF=X'02',
               PRIPROT=X'71',
               SECPROT=X'40',
               COMPROT=X'2000',
               PSERVIC=X'000000000000000000000200'
        OR
               PSERVIC=X'00000000000018502B507F00' Others
               PSERVIC=X'00000000000018502B507E00' Model 2, no Altscreen

TERMINAL definition
*************************

AUTINSTNAME   ==> M3278A
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3278
INSERVICE     ==> YES
OPERSEC       ==> 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3278
GROUP         ==> PDATD
DEVICE        ==> 3270
TERMMODEL     ==> 2
LIGHTPEN      ==> YES
AUDIBLEALARM  ==> YES
UCTRAN        ==> YES
IOAREALEN     ==> 2000,2000
ERRLASTLINE   ==> YES
ERRINTENSIFY  ==> YES
USERAREALEN   ==> 32
ATI           ==> YES
TTI           ==> YES
AUTOCONNECT   ==> NO
LOGONMSG      ==> YES
BIND SENT BY CICS :    01020271 40200000 00000080 00000000
                       00000000 00000002 00009300 00300000

OR

BIND SENT BY CICS :    01020271 40200000 00000080 00000000
                       00000018 502B507F 00009300 00300000

OR

Real Model 2
BIND SENT BY CICS :    01020271 40200000 00000080 00000000
                       00000018 502B507E 00009300 00300000
******************************************************************
2) LOCAL SNA 3277/78/79 (without special features) LUTYPE2
******************************************************************

S32782   MODEENT LOGMODE=S32782,   SNA LUTYPE2 3270
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'3080',
               RUSIZES=X'8585',
               PSERVIC=X'028000000000185018507F00'

TERMINAL definition
*************************

AUTINSTNAME   ==> M32782
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T32782
INSERVICE     ==> YES
```

```
TYPETERM definition
*************************

TYPETERM       ==> T32782
GROUP          ==> PDATD
DEVICE         ==> LUTYPE2
TERMMODEL      ==> 2
LIGHTPEN       ==> YES
AUDIBLEALARM   ==> YES
UCTRAN         ==> YES
IOAREALEN      ==> 256,256
ERRLASTLINE    ==> YES
ERRINTENSIFY   ==> YES
USERAREALEN    ==> 32
ATI            ==> YES
TTI            ==> YES
LOGONMSG       ==> YES
DISCREQ        ==> YES
RECEIVESIZE    ==> 256
BUILDCHAIN     ==> YES

BIND SENT BY CICS :          010303B1 B0308000 0085C780 00028000
                             00000018 5018507F 00000000 00000000

*****************************************************************
3) 3770 BATCH LU (3777)
*****************************************************************

BATCH    MODEENT LOGMODE=BATCH,    3770 BATCH
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'7080',
               PSERVIC=X'01310C70E100D20000E100D0'

TERMINAL definition
*************************

AUTINSTNAME    ==> M3770
AUTINSTMODEL   ==> ONLY
GROUP          ==> PDATD
TYPETERM       ==> T3770
INSERVICE      ==> YES
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3770
GROUP         ==> PDATD
DEVICE        ==> 3770
SESSIONTYPE   ==> BATCHDI
PAGESIZE      ==> 12,80
DISCREQ       ==> YES
AUTOPAGE      ==> YES
RECEIVESIZE   ==> 256
SENDSIZE      ==> 256
IOAREALEN     ==> 256,2048
BUILDCHAIN    ==> YES
BRACKET       ==> YES
ATI           ==> YES
TTI           ==> YES
AUTOCONNECT   ==> NO
HORIZFORM     ==> YES
VERTFORM      ==> YES
LDCLIST       ==> LDC2

Needs LDC declaration in TCT :

LDC2   DFHTCT TYPE=LDC,LOCAL=INITIAL
       DFHTCT TYPE=LDC,LDC=BCHLU
       DFHTCT TYPE=LDC,LOCAL=FINAL

BIND SENT BY CICS :         010303B1 B0708000 00000080 0001310C
                            70E100D2 0000E100 D0000000 00000000

*******************************************************************
4) 6670 LUTYPE4
*******************************************************************

S6670    MODEENT LOGMODE=S6670,    6670 LUTYPE4
               TYPE=1,
               FMPROF=X'07',
               TSPROF=X'07',
               RUSIZES=X'8585',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'5081',
               PSERVIC=X'04A840A000A840A000000C00'

TERMINAL definition
*************************

AUTINSTNAME   ==> M6670
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T6670
INSERVICE     ==> YES
```

```
TYPETERM definition
**************************

TYPETERM      ==> T6670
GROUP         ==> PDATD
DEVICE        ==> LUTYPE4
BUILDCHAIN    ==> YES
DISCREQ       ==> YES
RECEIVESIZE   ==> 256
UCTRAN        ==> YES
IOAREALEN     ==> 256,4096
FORMFEED      ==> YES
HORIZFORM     ==> YES
VERTFORM      ==> YES
ATI           ==> YES
TTI           ==> YES
PAGESIZE      ==> 50,80
AUTOPAGE      ==> YES
LOGONMSG      ==> NO
LDCLIST       ==> LDC1

Needs LDC declaration in TCT :

LDCS     DFHTCT TYPE=LDC,LDC=SYSTEM
LDC1     DFHTCT TYPE=LDC,LOCAL=INITIAL
         DFHTCT TYPE=LDC,DVC=(BLUCON,01),PROFILE=DEFAULT,LDC=PC,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(BLUPRT,02),PROFILE=BASE,LDC=PP,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(BLUPRT,08),PROFILE=BASE,LDC=P8,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(BLUPRT,08),PROFILE=DEFAULT,LDC=DP,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(BLUPCH,03),PROFILE=JOB,LDC=PM,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(BLUPCH,03),PROFILE=DEFAULT,LDC=DM,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED1,04),PROFILE=WPRAW,LDC=P1,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED1,04),PROFILE=DEFAULT,LDC=D1,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED2,05),PROFILE=OII1,LDC=P2,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED2,05),PROFILE=DEFAULT,LDC=D2,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED3,06),PROFILE=OII2,LDC=P3,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,DVC=(WPMED4,07),PROFILE=OII3,LDC=P4,
               PGESIZE=(50,80),PGESTAT=AUTOPAGE
         DFHTCT TYPE=LDC,LOCAL=FINAL

BIND SENT BY CICS :         010707B1 B0508100 00858580 0004A840
                            A000A840 A000000C 00000000 00000000
```

```
*****************************************************************
5) 3790 FULL FUNCTION LU
*****************************************************************

S3790A   MODEENT LOGMODE=S3790A,   3790 FULL FUNCTION LU
               TYPE=1,
               FMPROF=X'04',
               TSPROF=X'04',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               RUSIZES=X'8585',
               COMPROT=X'7080'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3790A
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3790A
INSERVICE     ==> YES

TYPETERM definition
*************************

TYPETERM      ==> T3790A
GROUP         ==> PDATD
DEVICE        ==> 3790
SENDSIZE      ==> 256
RECEIVESIZE   ==> 256
SESSIONTYPE   ==> USERPROG
BRACKET       ==> YES
IOAREALEN     ==> 256
ATI           ==> YES
TTI           ==> YES

BIND SENT BY CICS :         010404B1 B0708000 00858580 00000000

*****************************************************************
6) 3790 BATCH DATA INTERCHANGE
*****************************************************************

S3790B   MODEENT LOGMODE=S3790B,   3790 BATCH
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'04',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'7080',
               RUSIZES=X'8585',
               PSERVIC=X'013118400000920000E10050'
```

```
TERMINAL definition
*************************

AUTINSTNAME   ==> M3790B
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3790B
INSERVICE     ==> YES
TERMPRIORITY  ==> 50

TYPETERM definition
*************************

TYPETERM      ==> T3790B
GROUP         ==> PDATD
DEVICE        ==> 3790
SESSIONTYPE   ==> BATCHDI
AUTOPAGE      ==> YES
BUILDCHAIN    ==> YES
OBOPERID      ==> YES
IOAREALEN     ==> 256,2048
RELREQ        ==> YES
SENDSIZE      ==> 256
RECEIVESIZE   ==> 256
ATI           ==> YES
TTI           ==> YES
LDCLIST       ==> LDC2

Needs LDC declaration in TCT :

LDC2    DFHTCT TYPE=LDC,LOCAL=INITIAL
        DFHTCT TYPE=LDC,LDC=BCHLU
        DFHTCT TYPE=LDC,LOCAL=FINAL

BIND SENT BY CICS :          010304B1 B0708000 00858580 00013118
                             40000092 0000E100 50000000 00000000

*****************************************************************
7) 3790 SCSPRT
*****************************************************************

S3790C   MODEENT LOGMODE=S3790C,   3790 WITH SCS
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'3080',
               RUSIZES=X'8585',
               PSERVIC=X'010000000000000000000000'
```

```
TERMINAL definition
*************************

AUTINSTNAME   ==> M3790C
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3790C
INSERVICE     ==> YES

TYPETERM definition
*************************

TYPETERM      ==> T3790C   Note that CEDA changes DEVICE=3790,
GROUP         ==> PDATD    SESSIONTYPE=SCSPRT to DEVICE=SCSPRINT,
DEVICE        ==> 3790     SESSIONTYPE=blanks, PRINTERTYPE=3284.
SESSIONTYPE   ==> SCSPRT
BRACKET       ==> YES
SENDSIZE      ==> 256
RECEIVESIZE   ==> 256
ATI           ==> YES
TTI           ==> YES

BIND SENT BY CICS :         010303B1 B0308000 00858580 00010000

*****************************************************************
8) 3767 INTERACTIVE (FLIP-FLOP) LU
*****************************************************************

S3767    MODEENT LOGMODE=S3767,    3767 INTERACTIVE
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               PSERVIC=X'010000000000000000000000'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3767
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TERMPRIORITY  ==> 60
TYPETERM      ==> T3767
INSERVICE     ==> YES
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3767
GROUP         ==> PDATD
DEVICE        ==> 3767
VERTFORM      ==> YES
HORIZFORM     ==> YES
RELREQ        ==> YES
DISCREQ       ==> YES
IOAREALEN     ==> 256
AUTOPAGE      ==> NO
PAGESIZE      ==> 12,80
ATI           ==> YES
TTI           ==> YES
BRACKET       ==> YES
RECEIVESIZE   ==> 256
SENDSIZE      ==> 256

BIND SENT BY CICS :         010303B1 90308000 00000080 00010000
******************************************************************
9) 3650 INTERPRETER LU
     (SESTYPE = USERPROG  BRACKET = YES)
******************************************************************

S3650A   MODEENT LOGMODE=S3650A,   3650 SESTYPE=USERPROG
               TYPE=1,             BRACKET=YES
               FMPROF=X'04',
               TSPROF=X'04',
               PRIPROT=X'31',
               SECPROT=X'30',
               COMPROT=X'6000'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3650A
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3650A
INSERVICE     ==> YES
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3650A
GROUP         ==> PDATD
DEVICE        ==> 3650
SESSIONTYPE   ==> USERPROG
ROUTEDMSGS    ==> SPECIFIC
FMHPARM       ==> YES
RELREQ        ==> YES
DISCREQ       ==> YES
BRACKET       ==> YES
RECEIVESIZE   ==> 256
IOAREALEN     ==> 256,256
ATI           ==> YES
TTI           ==> YES
AUTOCONNECT   ==> NO

BIND SENT BY CICS :         01040431 30600000 00000080 00000000

*****************************************************************
10) 3650 HOST CONVERSATIONAL (3270) LU
*****************************************************************

S3650B   MODEENT LOGMODE=S3650B,   3650 SESTYPE=3270
               TYPE=1,             AND SESTYPE=3653
               FMPROF=X'04',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'6000'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3650B1
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3650B1
INSERVICE     ==> YES

TYPETERM definition
*************************

TYPETERM      ==> T3650B1
GROUP         ==> PDATD
DEVICE        ==> 3650
OBFORMAT      ==> YES
SESSIONTYPE   ==> 3270
RELREQ        ==> YES
DISCREQ       ==> YES
IOAREALEN     ==> 256
BRACKET       ==> YES
RECEIVESIZE   ==> 240
ATI           ==> NO
TTI           ==> YES

BIND SENT BY CICS :         010403B1 90600000 00000080 00000000
```

```
*******************************************************************
11) 3650 HOST CONVERSATIONAL (3653) LU
     (N.B. LOGMODE SAME AS HC (3270) LU)
*******************************************************************

S3650B   MODEENT LOGMODE=S3650B,   3650 SESTYPE=3270
               TYPE=1,             AND SESTYPE=3653
               FMPROF=X'04',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'6000'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3650B2
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3650B2
INSERVICE     ==> YES

TYPETERM definition
*************************

TYPETERM      ==> T3650B2
GROUP         ==> PDATD
DEVICE        ==> 3650
SESSIONTYPE   ==> 3653
RELREQ        ==> YES
DISCREQ       ==> NO
BRACKET       ==> YES
IOAREALEN     ==> 256
RECEIVESIZE   ==> 240
ROUTEDMSGS    ==> NONE
ATI           ==> NO
TTI           ==> YES

BIND SENT BY CICS :         010403B1 90600000 00000080 00000000

*******************************************************************
12) 3650 HOST COMMAND PROCESSOR LU
     (SESTYPE = USERPROG  BRACKET = NO)
*******************************************************************

S3650C   MODEENT LOGMODE=S3650C,   3650 SESTYPE=USERPROG
               TYPE=1,             BRACKET=NO
               FMPROF=X'04',
               TSPROF=X'04',
               PRIPROT=X'30',
               SECPROT=X'30',
               COMPROT=X'4000'
```

```
TERMINAL definition
*************************

AUTINSTNAME    ==> M3650C
AUTINSTMODEL   ==> ONLY
GROUP          ==> PDATD
TYPETERM       ==> T3650C
INSERVICE      ==> YES

TYPETERM definition
*************************

TYPETERM       ==> T3650C
GROUP          ==> PDATD
DEVICE         ==> 3650
SESSIONTYPE    ==> USERPROG
BRACKET        ==> NO
RELREQ         ==> NO
DISCREQ        ==> NO
RECEIVESIZE    ==> 256
IOAREALEN      ==> 256
ATI            ==> YES
TTI            ==> YES

BIND SENT BY CICS :          01040430 30400000 00000080 00000000

*******************************************************************
13) 8815 SCANMASTER (LU6.2 SINGLE SESSION)
*******************************************************************

SIN62  MODEENT LOGMODE=SIN62,      8815 SCANMASTER.
               TYPE=0,
               FMPROF=X'13',
               TSPROF=X'07',
               PRIPROT=X'B0',
               SECPROT=X'B0',
               COMPROT=X'50B1',
               PSNDPAC=X'00',
               SRCVPAC=X'00',
               SSNDPAC=X'00',
               RUSIZES=X'8585',
               PSERVIC=X'060200000000000000002C00'

TERMINAL definition
*************************

AUTINSTNAME  ==> MLU62
AUTINSTMODEL ==> ONLY
GROUP        ==> PDATD
TYPETERM     ==> SINLU62
INSERVICE    ==> YES
```

```
TYPETERM definition
*************************

TYPETERM     ==> SINLU62
GROUP        ==> PDATD
DEVICE       ==> APPC
RECEIVESIZE  ==> 2048
SENDSIZE     ==> 2048
ATI          ==> YES
TTI          ==> YES
Note: There is no RDO keyword equivalent of the MACRO
keyword 'FEATURE=SINGLE', because this is assumed with
RDO DEFINE TYPETERM when DEVICE=APPC.

BIND SENT BY CICS :         001307B0 B050B100 00858580 00060200
                            00000000 0000002C 00000800 00000000
                            0000001D 00090240 40404040 40404009
                            03006765 71D98A6C 300704C3 C9C3E2E6
                            F1000000 00000000 00000000 00000000

******************************************************************
14) 3290 (SDLC)
******************************************************************

S3290    MODEENT LOGMODE=S3290,    3290 SDLC
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'8787',
               PSERVIC=X'02800000000018503EA07F00'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3290
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TYPETERM      ==> T3290
INSERVICE     ==> YES
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3290
GROUP         ==> PDATD
DEVICE        ==> LUTYPE2
TERMMODEL     ==> 2
ALTSCREEN     ==> 62,160
DEFSCREEN     ==> 24,80
AUDIBLEALARM  ==> YES
UCTRAN        ==> YES
IOAREALEN     ==> 2000,2000
ERRLASTLINE   ==> YES
ERRINTENSIFY  ==> YES
USERAREALEN   ==> 32
ATI           ==> YES
TTI           ==> YES
LOGONMSG      ==> YES
ERRHILIGHT    ==> BLINK
RECEIVESIZE   ==> 1024

BIND SENT BY CICS :         010303B1 90308000 00878780 00028000
                            00000018 503EA07F 00000000 00000000

******************************************************************
15) 3601 WITH A 3604 ATTACHED
******************************************************************

S3600    MODEENT LOGMODE=S3600,    3601
               TYPE=1,
               FMPROF=X'04',
               TSPROF=X'04',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'7000',
               RUSIZES=X'0000'

TERMINAL definition
*************************

AUTINSTNAME   ==> M3600
AUTINSTMODEL  ==> ONLY
GROUP         ==> PDATD
TERMPRIORITY  ==> 50
TYPETERM      ==> T3600
INSERVICE     ==> YES
```

```
TYPETERM definition
*************************

TYPETERM      ==> T3600
GROUP         ==> PDATD
DEVICE        ==> 3600
AUTOPAGE      ==> NO
PAGESIZE      ==> 6,40
RELREQ        ==> YES
DISCREQ       ==> NO
IOAREALEN     ==> 256
SENDSIZE      ==> 224
RECEIVESIZE   ==> 256
USERAREALEN   ==> 100
ATI           ==> NO
TTI           ==> YES
BRACKET       ==> YES
LDCLIST       ==> BMSLLDC1
Needs LDC declaration in TCT :

BMSLLDC1 DFHTCT TYPE=LDCLIST,
               LDC=(DS,JP,PB=5,LP,MS)
         DFHTCT TYPE=LDC,
               LDC=(DS=1),
               DVC=3604,
               PGESIZE=(6,40),
               PGESTAT=PAGE
         DFHTCT TYPE=LDC,LDC=SYSTEM
BIND SENT BY CICS :          010404B1 B0700000 00000080 00000000
```

```
**********************************************************
16) LUTYPE 2 - Model 2 -  X'7E' in last byte of PSERVIC
**********************************************************
SP32772  MODEENT LOGMODE=SPECLU2,  (NO QUERY)
               TYPE=1,             Only Type Recognized
               FMPROF=X'03',       SNA
               TSPROF=X'03',       Non-SNA
               PRIPROT=X'B1',      Primary Protocol
               SECPROT=X'B0',      Secondary Protocol
               COMPROT=X'3080',    Common Protocol
               RUSIZES='85C7',
               PSERVIC=X'020000000000185000007E00'
```

TERMINAL Definition

```
AUTINSTNAME   ==> SPLU20
AUTINSTMODEL  ==> ONLY
GROUP         ==> SPDATD
TYPETERM      ==> SPLU2
```

TYPETERM Definition

```
TYPETERM      ==> SPLU2
GROUP         ==> SPDATD
DEVICE        ==> LUTYPE2
DEFSCREEN     ==> (0,0)
ALTSCREEN     ==> (0,0)
TERMMODEL     ==> 2
EXTENDEDDS    ==> NO
QUERY         ==> NO
RECEIVESIZE   ==> 256
SENDSIZE      ==> 1536

BIND Built by CICS     010303B1 B0308000 0085C780 00020000
                       00000018 5000007E 00009300 00300000

BIND SENT BY CICS :          010404B1 B0700000 00000080 00000000
```

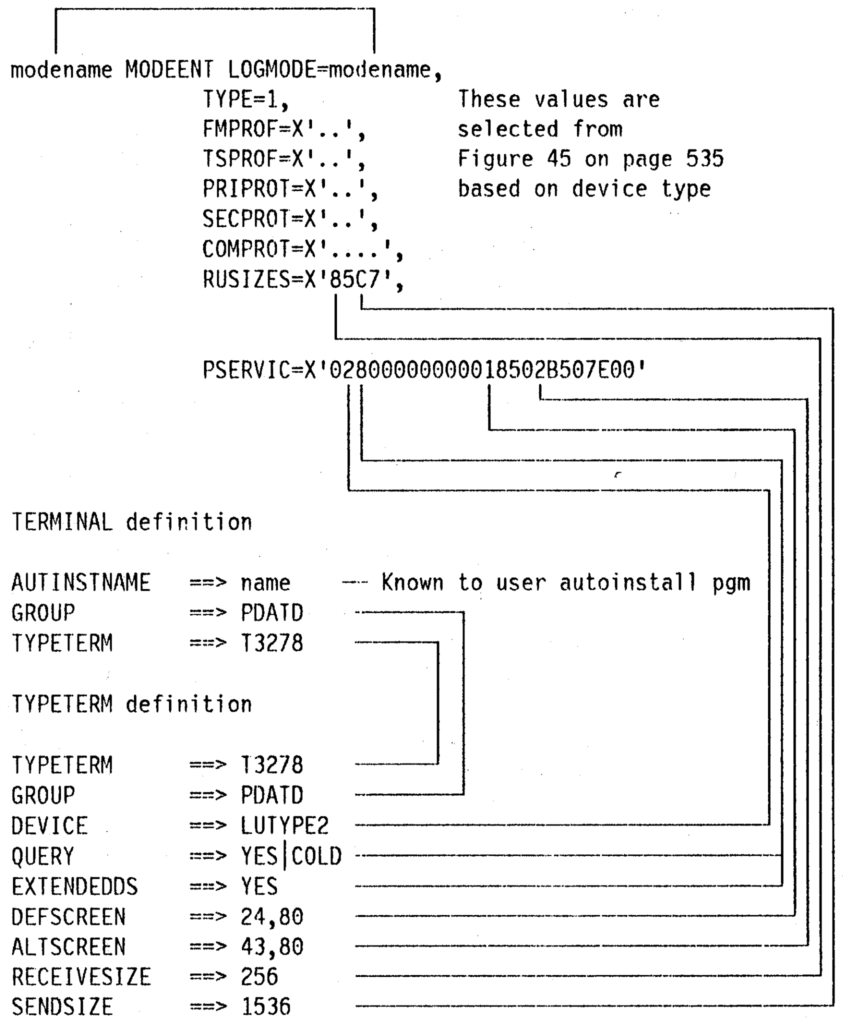
Using the X'7E' PSERVIC value for LUTYPE2 devices

The technique shown in the figure below is valid only when ALTSCREEN(0,0) and QUERY(NO) are also specified.

LUTYPE 2					
VTAM			CICS		
MODEL	DEFSCREEN	ALTSCREEN	MODEL	DEFSCREEN	ALTSCREEN
X'00'	X'0C28'	X'0000'	X'01'	IMPLIED 12 BY 40	0,0
	X'1850'	X'0000'	X'02'	IMPLIED 24 BY 80	0,0
X'7E'	X'0C28'	X'0000'	X'01'	IMPLIED 12 BY 40	0,0
	X'1850'	X'0000'	X'02'	IMPLIED 24 BY 80	0,0
	X'NNNN'	X'0000'	X'00'	X'NNNN'	0,0

Figure 46. Using the X'7E' PSERVIC value for LUTYPE2 devices. The two X'NNNN's in this figure must be equal.

Relationship between TERMINAL, TYPETERM, and MODEENT values



```
modename MODEENT LOGMODE=modename,
                TYPE=1,             These values are
                FMPROF=X'..',       selected from
                TSPROF=X'..',       Figure 45 on page 535
                PRIPROT=X'..',      based on device type
                SECPROT=X'..',
                COMPROT=X'....',
                RUSIZES=X'85C7',

                PSERVIC=X'028000000000185028507E00'

TERMINAL definition

AUTINSTNAME   ==> name    -- Known to user autoinstall pgm
GROUP         ==> PDATD
TYPETERM      ==> T3278

TYPETERM definition

TYPETERM      ==> T3278
GROUP         ==> PDATD
DEVICE        ==> LUTYPE2
QUERY         ==> YES|COLD
EXTENDEDDS    ==> YES
DEFSCREEN     ==> 24,80
ALTSCREEN     ==> 43,80
RECEIVESIZE   ==> 256
SENDSIZE      ==> 1536
```

LOGMODE definitions for CICS-supplied autoinstall models

This section contains LOGMODE definitions which match the supplied model TYPETERM/TERMINAL autoinstall definitions. The name of the LOGMODE entry is that of the matching TYPETERM definition.

```
DFHLU3   MODEENT LOGMODE=DFHLU3,   LU TYPE 3 PRINTER.
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'3080',
               RUSIZES=X'8585',
               PSERVIC=X'038000000000000000000200'

DFHSCSP  MODEENT LOGMODE=DFHSCSP,  LU TYPE 1 SCS PRINTER
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'7080',
               RUSIZES=X'8585',
               PSERVIC=X'010000010000000000000000'

DFHLU62T MODEENT LOGMODE=DFHLU62T, LU6.2 SINGLE-SESSION
               TYPE=0,
               FMPROF=X'13',
               TSPROF=X'07',
               PRIPROT=X'B0',
               SECPROT=X'B0',
               COMPROT=X'50B1',
               RUSIZES=X'8888',
               PSERVIC=X'060200000000000000002C00'

DFH3270  MODEENT LOGMODE=DFH3270,  3270
               TYPE=1,
               FMPROF=X'02',
               TSPROF=X'02',
               PRIPROT=X'71',
               SECPROT=X'40',
               COMPROT=X'2000',
               RUSIZES=X'F700'

DFH3270P MODEENT LOGMODE=DFH3270P, 3284/3286 BISYNC 3270P (QUERY)
               TYPE=1,
               FMPROF=X'02',
               TSPROF=X'02',
               PRIPROT=X'71',
               SECPROT=X'40',
               COMPROT=X'2000',
               RUSIZES=X'F700'
```

```
DFHLU2   MODEENT LOGMODE=DFHLU2,   SNA LUTYPE2 3270
               TYPE=1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'B0',
               COMPROT=X'3080',
               RUSIZES=X'85C7',
               PSERVIC=X'028000000000000000000300'
```

Glossary

This glossary includes definitions developed by the American National Standards Institute (ANSI). This material is reproduced from the American National Dictionary for Information Processing, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. ANSI definitions are preceded by an asterisk (*).

This glossary is also intended to give definitions of the most frequently used abbreviations. The *CICS/MVS Problem Determination Guide* (which gives descriptions, in alphabetic order, of CICS storage areas) may also be useful to the reader seeking explanations of abbreviated names.

No attempt is made here to define the abbreviations used in CICS commands and macros.

abend. Abnormal end of task.

ACB. Access method control block (VTAM and VSAM).

ACF. Advanced communication facility.

ACP. Abnormal condition program.

ACT. Application control table (DL/I).

AFCB. Authorization facility control block.

AID. Automatic initiate descriptor (CICS) *or* attention identifier.

ALT. Application load table.

ANS. American National Standard.

ANSI. American National Standards Institute.

APAR. Authorized program analysis report.

APF. Authorized program facility.

* **ASCII**. (American National Standard Code for Information Interchange) The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ATI. Automatic task initiation.

BDAM. Basic direct access method.

BFP. Built-in functions program.

BGAM. Basic graphics access method

BLL. Base locator linkage (COBOL).

BMP. Batch message processing program (IMS).

BMS. Basic mapping support.

BSAM. Basic sequential access method.

BSC. Binary synchronous communication.

BTAM. Basic telecommunications access method.

CAVM. CICS availability manager, the program that coordinates the processing of the active and alternate systems in an XRF environment.

CCB. Command control block.

CCE. Console control element.

COBOL. Common business-oriented language. A business data processing language.

command. In CICS, an instruction similar in format to a high-level programming language statement. Contrast with macro. CICS commands invariably include the verb EXECUTE (or EXEC).

CRH. Channel reconfiguration hardware.

CSA. Common storage area.

CSD. CICS system definition file.

CSKP. CICS activity keypoint program (DFHAKP).

CSMT. CICS-supplied transaction provided by the master terminal.

CSO. Control system operational group.

CSS. Control system service group.

CSTE. Destination used by the terminal abnormal condition group (DFHTACP).

CU. Operand of the terminal control table specifying the control unit attached to the channel.

DAM. Direct access method.

DASD. Direct access storage device.

DBD. Database definition.

DB/DC. Database/data communication.

DBP. Dynamic backout program.

DBR. Dynamic backout record.

DCA. Dispatch control area.

DCB. Data control block (operating system).

DCP. Dump control program.

DCT. Destination control table.

DEB. Data extent block.

DECB. Data event control block (operating system).

DFC. Data flow control.

DIP. Data interchange program.

DL/I. Data language/I.

DMB. Data management block (DL/I).

DMH. Device message handler.

DSA. Dynamic storage area.

DSECT. Dummy section defining a CICS data area.

DTB. Dynamic transaction backout.

DTP. Distributed transaction processing.

DWE. Deferred work element.

* **EBCDIC**. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block (operating system).

EDF. Execution (command level) diagnostic facility.

EIB. EXEC interface block.

EIP. EXEC interface program.

EIS. EXEC interface storage.

EMP. Event monitoring point.

EOT. End of transmission.

EREP. Error recording, editing, and printing.

ESDS. Entry-sequenced data set (VSAM).

ESE. Error status element.

ESTAE. Extended specify task asynchronous exit.

ETX. End-of-text character.

EXEC. EXECUTE (as used in a CICS command).

EXP. Command (EXEC) language translator program.

FAQE. Free area queue element.

FBA. Fixed block architecture.

FBWA. File browse work area.

FCP. File control program.

FCT. File control table.

FIOA. File input/output area.

FME. Function management end.

FMH. Function management header (SNA).

FWA. File work area.

GAM. Graphics access method.

GAP. Graphics attention program.

GTF. Generalized trace facility.

HLL. High-level language.

HLPI. High-level programming interface (CICS command-level interface).

HPO. High-performance option.

ICE. Interval control element.

ICP. Interval control program.

ICR. Independent component release.

ID. Identity.

IMS. Information Management System.

Intercommunication facilities. A generic term covering ISC and MRO.

I/O. Input/output (primarily from and to terminals).

IRC. (1) Interregion communicator. (2) Interregion communication access method for DL/I shared database support and for region remote connections with MRO.

IRS. Interchange record separator.

ISA. Initial storage area (PL/I).

ISB. Interface scheduling block (DL/I).

ISC. Intersystem communication - communication between separate systems by means of SNA networking facilities.

JC. Journal control.

JCA. Journal control area.

JCL. Job control language.

JCP. Journal control program.

JCR. Journal control record.

JCT. Journal control table.

JCTTE. Journal control table table entry.

journal. A set of one or more data sets to which records are written in chronological sequence during a CICS run, including the system log

KB. Kilobyte. 1024 bytes.

KCP. Task control program.

keypoint. A point at which the system status is recorded.

KP. Keypoint.

KSDS. Key-sequenced data set.

LDC. Logical device code.

LECB. Line event control block.

LERAD. Logical error address.

LIFO. Last-in/first-out (storage).

LIOA. Line input/output area.

LLA. Load list area.

LUW. Logical unit of work.

macro. In CICS, an instruction (similar in format to an assembler language instruction) that causes CICS to process a predefined set of statements called a macro definition. (Contrast with **command**.)

map. In CICS, a format established for a page or a portion of a page.

master terminal operator. Any CICS operator whose security key(s) allow use of the master terminal functions transaction.

MB. Megabyte. 1 048 576 bytes.

MCB. Message control block.

MCP. Message control program.

MCR. Message control record.

MCT. Monitoring control table.

MH. Message handler.

MRO. Multiregion operation. Communication between CICS systems in the same processor without the use of SNA networking facilities.

MSG. Message.

MTP. Master terminal program.

multiregion operation. Communication between CICS systems in the same processor without the use of SNA networking facilities.

NACP. Node abnormal condition program.

NCP. Network control program.

NEB. Node error block.

NET. Node error table.

NIB. Node initialization block (VTAM).

NL. New line.

NLT. Nucleus load table.

OLTEP. Online Test Executive Program.

OSPWA. Output services processor work area.

OS. Operating System.

PAM. Page allocation map.

part. With reference to journals, the set of records contained in one data set of the journal, and covering a determinable interval of time.

PCB. DL/I-CICS program communication block.

PCLOCK. Operand of the monitoring control table, specifying that the clock specified by number is to be stopped.

PCT. Program control table. The table defines the transactions known to the system.

PEP. Program error program (usually user-written).

PGT. Program global table (COBOL).

PGM. Program.

PIE. Program interrupt element (operating system).

* **PL/I**. A programming language designed for use in a wide range of commercial and scientific applications.

PLT. Program list table.

PPT. Program processing table. Defines all the application programs and maps in the system, and also various CICS modules and tables.

processor. Host processing unit.

PSB. Program specification block (DL/I).

PSW. Program status word.

PTF. Program temporary fix.

QEA. Queue element area.

QEC. Quiesce-at-end-of-chain.

RACF. The Resource Access Control Facility program product.

region. A section of the dynamic area that is allocated to a job step or system task. In this manual, the term is used to cover address spaces as well as regions.

RH. Request/response header.

RLN. Relative line number.

RMSR. Recovery management support recording (BTAM).

* **rollback**. A programmed return to a prior checkpoint.

RPL. Request parameter list.

RRN. Reached recovery node.

RSA. Register save area.

RU. Request unit.

SAA. Storage accounting area.

SAM. Sequential access method.

SCLOCK. Operand of the monitoring control table, specifying that the clock specified by number is to be started.

SCP. Storage control program.

SCS. SNA character stream.

SDLC. Synchronous data link control.

SDT. Series definition table.

Service Level Reporter II. A data reduction and analysis program product (program number 5740-DC3). Useful for analyzing CICS operating statistics.

SIT. System initialization table.

SLR. Service Level Reporter.

SMF. System management facilities.

SMI. Standard message indicator.

SNA. Systems network architecture.

SNT. Sign-on table.

SOS. Short on storage.

SPIE. Specify program interrupt element.

SRB. Service request block (MVS).

SRP. System recovery program.

SRT. System recovery table.

STAE. Specify task asynchronous exit.

starter system. A set of pregenerated programs provided as part of the CICS program product.

supervisory terminal operator. Any CICS operator whose security key(s) allow use of the supervisory terminal functions.

SVC. Supervisor call.

TACB. Transaction abend control block.

TACLE. Terminal abnormal condition line entry.

task. (1) A unit of work for the processor; therefore the basic multiprogramming unit under the control program. (CICS runs as a task under MVS.) (2) Under CICS, the execution of a transaction for a particular user. Contrast with transaction.

TBP. Transaction backout program.

TC. Terminal control.

TCA. Task control area.

TCAM. Telecommunications access method.

TCB. Task control block.

TCP. Terminal control program.

TCT. Terminal control table.

TCTLE. Terminal control table line entry.

TCTSE. Terminal control table system entry.

TCTTE. Terminal control table terminal entry.

TCU. Terminal control unit.

TDP. CICS transient data program.

TEB. Terminal error block.

TEP. Terminal error program.

TGT. Task global table (COBOL).

TIOA. Terminal input/output area.

TMA. Task monitoring area.

TP. Teleprocessing (subpool).

TR. Transaction restart.

transaction. A transaction may be regarded as a unit of processing (consisting of one or more application programs) initiated by a single request, often from a terminal. A transaction may require the initiation of one or more tasks for its execution. Contrast with task.

transaction identification code. Synonym for transaction identifier. For example, a group of up to four characters entered by an operator when selecting a transaction.

transaction identifier. Synonymous with transaction identification code.

TRP. CICS trace control program.

TS. Temporary storage.

TSP. CICS temporary storage program.

TST. Temporary storage table.

TSUT. Temporary storage unit table.

TTR. Track/record (disk address).

TWA. Transaction work area.

UEI. User exit interface.

UIB. User interface block (DL/I).

URL. User route list.

VS. Virtual storage.

VSAM. Virtual storage access method.

VSWA. VSAM work area.

VTAM. Virtual telecommunications access method.

WRE. Write request element.

WTO. Write-to-operator.

XA. Extended architecture

XLT. Transaction list table.

XRF. Extended recovery facility.

Index

Readers' Comments

CICS/MVS
Customization Guide
Version 2 Release 1 Modification 2

Publication No. SC33-0507-02

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments
SC33-0507-02

IBM®

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 6R1H
180 KOST ROAD
MECHANICSBURG PA 17055-0786

Fold and Tape **Please do not staple** Fold and Tape

C
A

Cu
Al

Customization Guide
SC33-0507-02

Version 2.1.2

CICS MVS

Program Number
5665-403

Printed in U.S.A.

SC33-0507-02