CICS® Transaction Server for VSE/ESA™

Problem Determination Guide

Release 1

GC33-1663-00

CICS® Transaction Server for VSE/ESA™

Problem Determination Guide

Release 1

GC33-1663-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 291.

First Edition (June 1999)

This edition applies to Release 1 of CICS Transaction Server for VSE/ESA, program number 5648-054, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The CICS for VSE/ESA Version 2.3 edition remains applicable and current for users of CICS for VSE/ESA Version 2.3.

Order publications through your IBM representative or the IBM branch office serving your locality.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make any comments, please use one of the methods described there.

© **Copyright International Business Machines Corporation 1979, 1999. All rights reserved.**
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface . . . vii
What this book is about . . . vii
Who this book is for . . . vii
What you need to know to understand this book . . . vii
How to use this book . . . vii
Notes on terminology . . . ix
Determining if a publication is current . . . xi

Part 1. Approach to problem determination . . . 1

Chapter 1. Introduction to problem determination . . . 3
Before you start—preliminary checks . . . 3
What to do next . . . 7

Chapter 2. Classifying the problem . . . 9
Classifying problems using symptom keywords . . . 9
Some more ways in which this manual classifies problems . . . 10
Using the symptoms to classify the problem . . . 10
Distinguishing between waits, loops, and poor performance . . . 14
Where to look next . . . 18

Chapter 3. Sources of information . . . 19
User documentation . . . 19
Manuals . . . 19
Source listings and link-edit maps . . . 19
Abend codes and error messages . . . 19
Symptom strings . . . 20
Change log . . . 20
Dumps . . . 20
Statistics . . . 21
Monitoring . . . 21
Transaction inputs and outputs . . . 22
Traces . . . 23
VSE service aids . . . 23

Part 2. Dealing with the problem . . . 25

Chapter 4. Dealing with transaction abends . . . 27
Collecting the evidence . . . 27
What the abend code can tell you . . . 28
CICS transaction abend codes . . . 28
Finding where a program check occurred . . . 30
Analyzing the problem further . . . 36
FEPI abends . . . 38

Chapter 5. Dealing with CICS system abends . . . 39
The documentation you need . . . 39
Interpreting the evidence . . . 40

© Copyright IBM Corp. 1979, 1999

Chapter 6. Dealing with waits 51
Techniques for investigating waits 52
How tasks are made to wait 59
Spooler waits 77
Transaction manager waits 78
Lock manager waits 82
Journal control waits 85
Task control waits 88
Storage waits 90
Temporary storage waits 91
Terminal waits 94
VTAM terminal control waits 102
Interregion and intersystem communication waits 104
Transient data waits 105
Loader waits 108
File control waits 108
Interval control waits 115
XRF alternate system waits 121
CICS system task waits 122
FEPI waits 123
What to do if CICS has stalled 124

Chapter 7. Dealing with loops 129
What sort of loop is indicated by the symptoms? 129
Investigating loops that cause transactions to abend with abend code AICA 134
Investigating loops that are not detected by CICS 136
What to do if you cannot find the reason for a loop 138

Chapter 8. Dealing with performance problems 139
Finding the bottleneck 140
Finding why tasks fail to get attached to the transaction manager 141
Finding why tasks fail to get attached to the dispatcher 142
Finding why tasks fail to get an initial dispatch 144
Finding out why tasks take a long time to complete 146
A summary of performance bottlenecks, symptoms, and causes 146

Chapter 9. Dealing with incorrect output 149
Trace output is incorrect 149
Dump output is incorrect 153
Wrong data has been displayed on a terminal 156
Incorrect data is present on a VSAM data set 163
An application did not work as expected 163
Your transaction produced no output at all 164
Your transaction produced some output, but it was wrong 169

Chapter 10. Dealing with storage violations 173
Avoiding storage violations 173
Two kinds of storage violation 174
CICS has detected a storage violation 174
Storage violations that affect innocent transactions 181
Programming errors that can cause storage violations 182
Storage recovery 183

Chapter 11. Dealing with XRF errors 185

Symptoms of problems in an XRF complex . . . 185
Debugging the overseer sample program . . . 189
XRF trace . . . 190
The CAVM data sets . . . 190

Chapter 12. External CICS interface . . . 193

Chapter 13. Dealing with MRO problems . . . 195

Part 3. Using traces and dumps in problem determination . . . 197

Chapter 14. Using traces in problem determination . . . 199
Normal CICS tracing . . . 207
CICS exception tracing . . . 220
CICS XRF tracing . . . 221
XRF trace entry types . . . 222
Program check and abend tracing . . . 224
VTAM buffer tracing . . . 224
Using FEPI trace . . . 224

Chapter 15. Using dumps in problem determination . . . 225
Controlling dump action . . . 225
Analyzing dumps . . . 236
Locating the last command or statement . . . 241
Locating program data . . . 242
Storage freeze . . . 252
Using FEPI dump . . . 252

Chapter 16. The global trap/trace exit . . . 253
Establishing the exit . . . 253
Information passed to the exit . . . 254
Actions the exit can take . . . 254
Program check handling . . . 255
Coding the exit . . . 255

Part 4. Working with IBM to solve your problem . . . 257

Chapter 17. IBM program support . . . 259
When to contact the Support Center . . . 259
Dealing with the Support Center . . . 259
IBM Program Support structure . . . 262
Reporting a FEPI problem to IBM . . . 263

Chapter 18. APARs, fixes, and PTFs . . . 265
The APAR process . . . 265
Collecting the documentation for the APAR . . . 265
Sending the documentation to the change team . . . 266
Applying the fix . . . 267

Part 5. Appendixes . . . 269

Appendix A. SDUMP contents and INFOANA EXIT keywords 271
Finding the control blocks from the keywords 272
Finding the keywords from the control blocks 278

Appendix B. Summary data for PG and US keywords 285
PG keyword 285
US keyword 289

Notices 291
Trademarks and service marks 292

Bibliography 293
Books from VSE/ESA 2.4 base program libraries 294
Books from VSE/ESA 2.4 optional program libraries 296

Index 299

Preface

What this book is about

This book helps you to determine the causes of problems in a system that uses CICS Transaction Server for VSE/ESA™. It contains information about resolving CICS® application and system problems, dealing with the IBM Support Center, and handling authorized program analysis reports (APARs).

In general, this book does not describe methods of problem determination for the CICS Transaction Server for VSE/ESA Front End Programming Interface. This information is included in the *CICS Front End Programming Interface User's Guide*.

Who this book is for

This book is for those responsible for debugging CICS systems and application programs.

What you need to know to understand this book

This book assumes that you have a good knowledge of CICS. If you are using the book to resolve system problems, you need to be familiar with the books that tell you how to install and use a CICS system.

How to use this book

The information in this book is mainly for reference. Use the information to first classify and then find a solution to your problem.

Refer to Table 1 to find the section of the book that you need to read.

Table 1 (Page 1 of 2). Road map for Problem Determination

If you want to...	**Refer to...**
Go through some preliminary checks	Chapter 1, "Introduction to problem determination" on page 3
Classify the problem according to its symptoms	Chapter 2, "Classifying the problem" on page 9
Look for information to help you diagnose the problem	Chapter 3, "Sources of information" on page 19
Resolve transaction abnormal terminations.	Chapter 4, "Dealing with transaction abends" on page 27
Resolve system abnormal terminations.	Chapter 5, "Dealing with CICS system abends" on page 39
Decide whether the problem is caused by a wait, a loop, or a performance problem.	"Distinguishing between waits, loops, and poor performance" on page 14
Resolve problems caused by waits.	Chapter 6, "Dealing with waits" on page 51
Resolve problems caused by loops.	Chapter 7, "Dealing with loops" on page 129
Resolve problems caused by performance problems.	Chapter 8, "Dealing with performance problems" on page 139
Know what to do if you don't get the output you expected.	Chapter 9, "Dealing with incorrect output" on page 149

© Copyright IBM Corp. 1979, 1999

Table 1 (Page 2 of 2). Road map for Problem Determination	
If you want to...	**Refer to...**
Resolve problems caused by storage violations.	Chapter 10, “Dealing with storage violations” on page 173
Resolve problems involving XRF.	Chapter 11, “Dealing with XRF errors” on page 185
Resolve problems with the external CICS interface.	Chapter 12, “External CICS interface” on page 193
Resolve problems with MRO.	Chapter 13, “Dealing with MRO problems” on page 195
Use CICS trace.	Chapter 14, “Using traces in problem determination” on page 199
Use CICS dump.	Chapter 15, “Using dumps in problem determination” on page 225
Report the problem to IBM.	Chapter 16, “The global trap/trace exit” on page 253
Look up a dump exit keyword.	Appendix A, “SDUMP contents and INFOANA EXIT keywords” on page 271

Notes on terminology

The terms listed in Table 2 are commonly used in the CICS Transaction Server for VSE/ESA Release 1 library. See the *CICS Glossary* for a comprehensive definition of terminology.

Table 2 (Page 1 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	**Definition (and abbreviation if appropriate)**
$(the dollar symbol)	In the character sets and programming examples given in this book, the dollar symbol ($) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.
BSM	BSM is used to indicate the basic security management supplied as part of the VSE/ESA product. It is RACROUTE-compliant, and provides the following functions: • Signon security • Transaction attach security
C	The C programming language
CICSplex	A CICSplex consists of two or more regions that are linked using CICS intercommunication facilities. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources accessed by the AORs
CICS Data Management Facility	The new CICS Transaction Server for VSE/ESA Release 1 facility to which all statistics and monitoring data is written, generally referred to as “DMF”
CICS/VSE	The CICS product running under the VSE/ESA operating system, frequently referred to as simply “CICS”
COBOL	The COBOL programming language
DB2 for VSE/ESA	Database 2 for VSE/ESA which was previously known as “SQL/DS”.

Table 2 (Page 2 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	**Definition (and abbreviation if appropriate)**
ESM	ESM is used to indicate a RACROUTE-compliant external security manager that supports some or all of the following functions: • Signon security • Transaction attach security • Resource security • Command security • Non-terminal security • Surrogate user security • MRO/ISC security (MRO, LU6.1 or LU6.2) • FEPI security.
FOR (file-owning region)—also known as a DOR (data-owning region)	A CICS region whose primary purpose is to manage VSAM and DAM files, and VSAM data tables, through function provided by the CICS file control program.
IBM C for VSE/ESA	The Language Environment version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment version of the PL/I programming language compiler. Generally referred to as “PL/I VSE”.
IBM Language Environment for VSE/ESA	The common runtime interface for all LE-conforming languages. Generally referred to as “LE/VSE”.
PL/I	The PL/I programming language
VSE/POWER	Priority Output Writers Execution processors and input Readers. The VSE/ESA spooling subsystem which is exploited by the report controller.
VSE/ESA System Authorization Facility	The new VSE facility which enables the new security mechanisms in CICS TS for VSE/ESA R1, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both the printed hardcopy and the BookManager softcopy versions of a publication are in step, but subsequent updates are normally made available in softcopy before they appear in hardcopy.

For CICS Transaction Server for VSE/ESA Release 1 books, softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx and on the *VSE/ESA Collection Kit* CD-ROM, SK2T-0060-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-20 is more up-to-date than SK2T-0730-19. The collection kit is also clearly dated on the front cover.

For individual books, the suffix number is incremented each time it is updated, so a publication with order number SC33-0667-02 is more recent than one with order number SC33-0667-01. Updates in the softcopy are clearly marked by revision codes (usually a “#” character) to the left of the changes.

Note that book suffix numbers are updated as a product moves from release to release, as well as for updates within a given release. Also, the date in the edition notice is not changed until the hardcopy is reissued.

Part 1. Approach to problem determination

Part 1 contains:

Chapter 1. Introduction to problem determination 3
Before you start—preliminary checks 3
What to do next 7

Chapter 2. Classifying the problem 9
Classifying problems using symptom keywords 9
Some more ways in which this manual classifies problems 10
Using the symptoms to classify the problem 10
Distinguishing between waits, loops, and poor performance 14
Where to look next 18

Chapter 3. Sources of information 19
User documentation 19
Manuals 19
Source listings and link-edit maps 19
Abend codes and error messages 19
Symptom strings 20
Change log 20
Dumps 20
Statistics 21
Monitoring 21
Transaction inputs and outputs 22
Traces 23
VSE service aids 23

© Copyright IBM Corp. 1979, 1999

Chapter 1. Introduction to problem determination

This book tells you how to find the reasons for problems with your CICS system. This chapter contains the following topics:

- "Before you start—preliminary checks"
- "What to do next" on page 7

Usually, you start with a symptom, or set of symptoms, and trace them back to their cause. This book describes tools and techniques you can use to find the cause of a problem and suggests action for solving the problem.

Sometimes, you cannot solve the problem yourself if, for example, it is caused by limitations in the hardware or software you are using. If the cause of the problem is CICS code, you need to contact IBM®, as described in Part 4, "Working with IBM to solve your problem" on page 257.

Before you start—preliminary checks

Before you go further into looking for the cause of the problem, run through the following preliminary checks. These checks might highlight a simple cause or, at least, narrow the range of possible causes.

As you go through the questions, make a note of anything that might be relevant to the problem. Even if the observations you record do not at first suggest a cause, they could be useful to you later if you need to carry out systematic problem determination.

1. **Has the CICS system run successfully before?**

   If the CICS system has not run successfully before, it is possible that you have not yet set it up correctly. You need to read other books in the CICS library for guidance on doing this.

   If you are currently migrating to CICS Transaction Server for VSE/ESA Release 1, ensure that you are aware of all the changes that have been made for this version. For details of these, see the *CICS Migration Guide*.

2. **Are there any messages explaining the failure?**

   If a transaction abends, and the task terminates abnormally, CICS sends a message reporting the fact to the CSMT log (or your site replacement). If you find a message there, it might immediately suggest a reason for the failure.

   Were there any unusual messages associated with CICS start up, or while the system was running before the error occurred? These might indicate some system problem that prevented your transaction from running successfully.

   If you see any messages that you do not understand, look in the *VSE/ESA Messages and Codes Volume 3* manual for an explanation and, perhaps, a suggested course of action that you can take to resolve the problem.

3. **Can you reproduce the error?**

   __ a. Can you identify any application that is always in the system when the problem occurs?

      - Check for application coding errors.

© Copyright IBM Corp. 1979, 1999

- Check that you have defined sufficient resources for the application, such as VSAM file strings. Typically, if you had not defined sufficient resources, you would find that the problem is related to the number of users of the application.

__ b. Are you using exit programming interface (XPI) calls? If so, be sure to observe the XPI protocols and restrictions exactly. For programming information about the XPI, see the *CICS Customization Guide*.

The exit programming interface enables you to invoke a domain and enter its environment directly; using it incorrectly can cause severe CICS system problems. Here are some particular points for your attention:

- Are the input parameters correct? If their format is not valid, they are rejected by the called domain, and an exception trace is made. If their values are acceptable to the domain but inappropriate for the system, they could cause unpredictable effects.
- Be aware that you cannot use some XPI calls within some of the user exits. If you do, the results can be unpredictable, and can cause CICS to stall or abend. The *CICS Customization Guide* tells you which exits can use XPI calls and which cannot.

__ c. If the problem is not related to any particular application, have you considered your CICS system initialization parameters? Poorly defined parameters may be the cause of problems in your system. You can find guidance about setting up your CICS system in the *CICS System Definition Guide*.

__ d. Does the problem seem to be related to system loading? If so, the system might be running near its maximum capacity, or it might be in need of tuning. For guidance about dealing with this sort of problem, see the *CICS Performance Guide*.

4. **Does the failure occur at specific times of day?**

   If the failure occurs at specific times of day, it could be dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so those are the times when load-dependent failures are most likely to happen. However, if your CICS network extends across more than one time zone, peak system loading might occur at some other time of day.

5. **Is the failure intermittent?**

   If an error is intermittent, particularly if it does not show the same symptoms, the problem might be more difficult to resolve. An intermittent failure can be caused by a "random" storage overlay. Furthermore, the transaction that caused the error might have been deleted from the system long before the symptoms are seen.

   A method you can use to investigate random overlays is described in Chapter 10, "Dealing with storage violations" on page 173.

6. **Have any of the following changes been made since the last successful run?**

   Service

   __ a. Have you applied a PTF to CICS?

__ b. Did it install successfully or did you get an error message during installation? If you installed it successfully, check with IBM for any PTF error.

__ c. Have any patches applied to any other program affected the way CICS interfaces with the program?

Hardware

__ a. Have you changed any of your hardware?

Software

__ a. Have you changed any of your software?

__ b. If you have installed a new or modified application, check for error messages in the output from the following:

- Translator
- Compiler
- Assembler
- Linkage editor

Administration

__ a. Have you changed your initialization procedure, for example by JCL, CICS system initialization or override parameters, or VTAM® CONFIG/LIST?

__ b. Has CICS generated any error messages during initialization?

__ c. Have you installed any resource definitions defined using CEDA?

If the definitions were made but not installed when CICS was last terminated, they might not have been preserved over the termination and subsequent start up. In general, changes made to the CSD but not installed are not visible when the CICS system is warm started. However, if the change was in a group in a list specified on the GRPLIST system initialization parameter specified on a cold start, it is effectively installed during startup. (Changes which have been installed are not visible after a cold start unless they were made to a group in a list specified on the GRPLIST system initialization parameter.

If START=AUTO was specified in the system initialization table, or as an override, you need to examine the job log to find out how the CICS system last initialized.

For detailed guidance about the ways in which resources can be defined and installed, see the *CICS Resource Definition Guide*.

7. **Are specific parts of the network affected by the problem?**

__ a. Can you identify specific parts of the network that the problem affects? If you can, look for any explanatory message from the access method. Even if no message has been sent to the console, you might find one in the CSNE log.

__ b. Have you made any network-related changes?

__ c. If the problem affects a single terminal, are your terminal definitions correct? Consider both the RDO TERMINAL definition, and the RDO TYPETERM definition it uses.

__ d. If the problem affects a number of terminals, can you identify a factor that is common to all of them? For example:

- Do they use the same RDO TYPETERM definition? If so, it is likely that there is an error in that RDO TYPETERM definition.
- Is the whole network affected? If so, CICS has probably stalled. See “What to do if CICS has stalled” on page 124 for advice.

8. **Has the application run successfully before?**

__ a. Have any changes been made to the application since it last ran successfully?

Examine the new or modified part of the application.

__ b. Have you used RDO to create or alter a transaction, program, or mapset? You must install these definitions before the resources are available to the running CICS region.

__ c. If you changed any maps, have you created both a new phase (TYPE=MAP) and a new DSECT (TYPE=DSECT), and then recompiled every program using the new DSECT? Use the CEMT commands:

```
CEMT SET PROGRAM(mapset) NEWCOPY
CEMT SET PROGRAM(all programs) NEWCOPY
```

(See the *CICS-Supplied Transactions* manual for guidance about the CEMT transaction.)

__ d. Have all the functions of the application been fully exercised before?

Establish what the program was doing when the error occurred, and check the source code in that part of the program.

If a program has been run successfully on previous occasions, examine the contents of any records, screen data, and files that were being processed when the error occurred. They may contain some unusual data value that caused a rarely used path in the program to be invoked.

__ e. Check that the application successfully retrieved the records that it required at the time of the error.

__ f. Check that all fields within the records at the time of the error contain data in a format acceptable to the program. Use CICS dump to do this.

If you can reproduce the problem in a test system, you can use programming language debug tools and the CEDF transaction to check the data and solve the problem.

9. **The application has not run successfully before**

If your application has not yet run successfully, examine it carefully to see if you can find any errors in it.

__ a. Check the output from the translator, the compiler, and the linkage editor, for any reported errors.

If your application fails to translate, compile or assemble, or link-edit cleanly into the correct phase library, it will also fail to run if you attempt to invoke it.

__ b. Check the coding logic of the application. Do the symptoms of the failure indicate the function that is failing and, therefore, the piece of code in error?

__ c. The following is a list of some programming errors commonly found in applications:

- CICS areas are addressed incorrectly
- The rules for quasi-reentrancy are not followed
- Transient data is managed incorrectly
- File resources are not released
- Storage is corrupted by the program
- Return codes from CICS requests are ignored

See “File control waits” on page 108 for more information about common user errors in application programming.

What to do next

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you should now be able to resolve it, possibly with the help of other books in the CICS library and in the libraries of other licensed programs.

If you have not yet found the cause, you must start to look at the problem in greater detail. Begin by finding the best category for the problem, using the approach described in Chapter 2, “Classifying the problem” on page 9.

Chapter 2. Classifying the problem

The purpose of this chapter is to help you classify your problem into one of the categories used by the IBM Support Center for its service procedures. It contains the following topics:

- "Classifying problems using symptom keywords"
- "Some more ways in which this manual classifies problems" on page 10
- "Using the symptoms to classify the problem" on page 10
- "Distinguishing between waits, loops, and poor performance" on page 14
- "Where to look next" on page 18

IBM Support Center staff have found that classifying the problem first is a good approach to problem determination.

Classifying problems using symptom keywords

IBM keeps records of all known problems with its licensed programs on the RETAIN database. IBM Support Center staff continually update the database as new problems are reported, and they regularly search the database to see if problems they are told about are already known.

If you have the IBM INFORMATION/ACCESS licensed program, 5665-266, you can look on the RETAIN database yourself. Each of the problems there has a classification type. For software problems, the classifications you will see are:

- **ABEND** (for **transaction abends**, see Chapter 4, "Dealing with transaction abends" on page 27; for **system abends**, see Chapter 5, "Dealing with CICS system abends" on page 39)
- **WAIT** (see Chapter 6, "Dealing with waits" on page 51)
- **LOOP** (see Chapter 7, "Dealing with loops" on page 129)
- **POOR PERFORMANCE**, or **PERFM** (see Chapter 8, "Dealing with performance problems" on page 139)
- **INCORRECT OUTPUT**, or **INCORROUT** (see Chapter 9, "Dealing with incorrect output" on page 149)
- **MESSAGE**

All but the last of these, MESSAGE, are considered in this book. If you receive a CICS error message, look in the *VSE/ESA Messages and Codes Volume 3* manual for an explanation. If you get a message from another IBM program, or from the operating system, you need to look in the messages and codes book from the appropriate library for an explanation of what that message means.

The *VSE/ESA Messages and Codes Volume 3* manual might give you enough information to solve the problem quickly, or it might redirect you to this manual for further guidance. If you are unable to deal with the message, you may eventually need to contact the IBM Support Center for assistance.

One type of problem that might give rise to a number of symptoms, usually ill-defined, is that of poor application design. Checking the design of an application is beyond the scope of this book, but one instance is described in "Poor application

© Copyright IBM Corp. 1979, 1999

design” on page 17, where you will find an example of how bad design can give rise to an application with poor usability.

Some more ways in which this manual classifies problems

In addition to the RETAIN classifications used by the IBM Support Centers, this book considers the following types of problem to belong in classes of their own for the purpose of investigation:

- **Storage violations** (see Chapter 10, “Dealing with storage violations” on page 173)
- **XRF errors** (see Chapter 11, “Dealing with XRF errors” on page 185)
- **EXCI errors** (see Chapter 12, “External CICS interface” on page 193)
- **MRO errors** (see Chapter 13, “Dealing with MRO problems” on page 195)

Whereas XRF, EXCI, and MRO errors can easily be classified in a straightforward way, confirming that you have a storage violation can be difficult. Unless you get a CICS message stating explicitly that you have a storage violation, you could get almost any symptom, depending on what has been overlaid. You might, therefore, classify it initially as one of the RETAIN symptom types described in “Classifying problems using symptom keywords” on page 9.

Using the symptoms to classify the problem

The following paragraphs classify the problem on the basis of the symptoms you observe.

The symptoms might enable you to classify the problem correctly at once, but sometimes classification is not so straightforward. You may need to consider the evidence carefully before making your decision. You might need to make a “best guess”, and then be prepared to reconsider later on the basis of further evidence.

Look for the chapter heading that most nearly describes the symptoms you have, and then follow the advice given there.

CICS has stopped running

Consider, firstly, the possibility that CICS might still be running, but only slowly. Be certain that there is no activity at all before carrying out the checks in this section. If CICS *is* running slowly, you probably have a performance problem. If so, read “CICS is running slowly” on page 11 to confirm this before going on to Chapter 8, “Dealing with performance problems” on page 139 for advice about what to do next.

There are three main reasons why CICS might unexpectedly stop running:

1. There could be a CICS system abend.
2. CICS could be in a wait state. In other words, it could have stalled.
3. A program could be in a tight loop.

Look in the following places for any message that might explain the situation:

1. **The VSE console**. Look for any message saying that the CICS job has abnormally terminated. If you find one, it means that a CICS system abend

has occurred. In such a case, you should examine the CSMT log to see which abend message has been written there.

If you do not find any explanatory message on the VSE console, check in the CSMT log anyway.

2. **The CSMT log**. CSMT is the transient data destination to which abend messages are written. If you find a message there, look in the *VSE/ESA Messages and Codes Volume 3* manual to make sure there has been a CICS system abend.

   If you see only a transaction abend message in the CSMT log, that will not account for CICS itself not running, and you should not classify the problem as an abend. A faulty transaction could hold CICS up, perhaps indefinitely, but CICS would resume work again once the transaction abended.

Here are two examples of messages that might accompany CICS system abends, and which you would find on the CSMT log:

DFHST0001 *applid* **An abend (code** *aaa/bbbb***) has occurred at offset X'***offset***' in module** *modname*.

DFHSR0601 Program interrupt occurred with system task *taskid* **in control**

If you get either of these messages, or any others for which the system action is to terminate CICS, turn to Chapter 5, “Dealing with CICS system abends” on page 39 for advice on what to do next.

If you cannot find a message telling you that CICS has terminated, it is likely that the CICS system is in a wait state, or that some program is in a tight loop and not returning control to CICS. These two possibilities are dealt with in Chapter 6, “Dealing with waits” on page 51 and Chapter 7, “Dealing with loops” on page 129, respectively.

CICS is running slowly

If CICS is running slowly, it is likely that you have a performance problem. It could be because your system is badly tuned, or because it is operating near the limits of its capacity. You will probably notice that the problem is worst at peak system load times, typically at mid-morning and mid-afternoon. If your network extends across more than one time zone, peak system load might occur at some other time.

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly designed transaction could be the cause. Classify the problem initially as “poor performance”, but be prepared to reconsider your classification later.

The following are some symptoms that could contribute to your perception that CICS is running slowly:

- Tasks take a long time to start running.
- Some low priority tasks will not run at all.
- Tasks start running, but take a long time to complete.
- Some tasks start running, but do not complete.
- No output is obtained.
- Terminal activity is reduced, or has ceased.

Some of these symptoms do not, in isolation, necessarily mean that you have got a performance problem. They could indicate that some task is in a loop, or is waiting on a resource that is not available. Only you can judge whether what you see should be classified as “poor performance”, in the light of all the evidence you have.

You might be able to gather more detailed evidence by using the tools and techniques that CICS provides for collecting performance data. The following is a summary of what is available:

- **CICS statistics**
  You can use these to gather information about the CICS system as a whole, without regard to tasks.
- **CICS monitoring**
  You can use this facility to collect information about CICS tasks.
- **CICS tracing**
  This is not a specific tool for collecting performance data, but you can use it to gather detailed information about performance problems.

For guidance about using these tools and techniques, and advice about performance and system tuning in general, see the *CICS Performance Guide*.

You can find guidance about identifying specific performance bottlenecks in your CICS system in Chapter 8, “Dealing with performance problems” on page 139.

A task fails to start

If a task fails to start, look first in the CSMT and CSNE logs for any explanatory message. If you do not find one, the task is possibly being prevented from starting because either the system is running at the MXT limit (maximum number of tasks permitted in the CICS system), the transaction is queuing for admittance to a transaction class, (RDO TRANCLASS) or for other performance reasons.

Classify the problem provisionally as “poor performance”, and turn to Chapter 8, “Dealing with performance problems” on page 139 for further guidance.

A task is running slowly

If just one task is running slowly, it is likely that the explanation lies with the task itself. It could be in a loop, or it could periodically be entering a wait state. You need to decide which of these possibilities is the most likely before starting systematic problem determination. The ways that you might distinguish between waits and loops are described in “Distinguishing between waits, loops, and poor performance” on page 14.

Note: Do not overlook the possibility that the task might simply be doing unnecessary work that does not change the final result—for example, starting a skip sequential browse with large gaps between the keys, or failing to finish one because it is holding on to resources.

A task stops running at a terminal

When a task stops running at a terminal, you will notice either or both of these symptoms:

- No output is obtained at the terminal
- The terminal accepts no input

First, make sure that the task is still in the system. Use the CEMT INQUIRE TASK transaction to check its status, and make sure that it has not ended by simply writing back to the terminal.

If the terminal has a display unit, check to see whether a special symbol has been displayed in the operator information area that could explain the fault. If the operator information area is clear, next check to see that no message has been sent to any of the transient data destinations used for error messages, for example:

- CSMT, the destination for terminal error and abend messages
- CSTL, the destination for terminal I/O error messages
- CSNE, the destination for error messages written by DFHZNAC and DFHZNEP

For details of the destinations used by CICS, see the *CICS System Definition Guide*. If you can find no explanation for the problem, the fault is probably associated with the task running at the terminal. These are the possibilities:

- The task is in a wait state.
- The task is in a loop.
- There is a performance problem.

Read “Distinguishing between waits, loops, and poor performance” on page 14 to find out which of these is the most likely explanation. You can then read the appropriate chapter for advice about dealing with the problem.

A transaction has abended

If the transaction abended when you ran your application, CICS gives you an error message on your screen as well as a message on the CSMT log.

Look in the *VSE/ESA Messages and Codes Volume 3* manual for an explanation of the message, and, perhaps, advice about what you should do to solve the problem. If the code is not there, or the explanation or advice given is not sufficient for you to solve the problem, turn to Chapter 4, “Dealing with transaction abends” on page 27.

You have obtained incorrect output

Incorrect output is any sort of output that you were not expecting. However, use the term with care in the context of problem determination, because it might be a secondary effect of some other type of error. For example, looping could be occurring if you get any sort of repetitive output, even though that output is not what you had expected. Also, CICS responds to many errors that it detects by sending messages. You might regard the messages as “incorrect output”, but they are only symptoms of another type of problem.

If you have received an unexpected message, and its meaning is not at first clear, look in the *VSE/ESA Messages and Codes Volume 3* manual for an explanation.

These are the types of incorrect output that are dealt with in this book, in Chapter 9, “Dealing with incorrect output” on page 149.

- **Incorrect trace or dump data:**
  - Wrong destination
  - Wrong type of data captured
  - Correct type of data captured, but the data values were unexpected
- **Wrong data displayed on the terminal**

A storage violation has occurred

When CICS detects that storage has been corrupted, this message is sent to the console:

DFHSM0102 *applid* **A storage violation (code X'***code***') has been detected by module** *modname*.

If you see this message, or you know (through other means) that a storage violation has occurred, turn to Chapter 10, “Dealing with storage violations” on page 173 for advice about dealing with the problem.

In many cases storage violations go undetected by CICS, and you only find out that they have occurred when something else goes wrong as a result of the overlay. You could, for example, get a program check because code or data has been overlaid. You might suspect some other type of problem at first, and only after starting your investigation find that a storage violation has occurred.

You can avoid many storage violations by enabling storage protection (STGPROT=YES in the SIT) and command protection (CMDSEC=YES in the SIT). See the *CICS System Definition Guide* for more information about these system initialization parameters.

An XRF error has occurred

If an XRF error has occurred, turn to Chapter 11, “Dealing with XRF errors” on page 185 for advice.

Distinguishing between waits, loops, and poor performance

Waits, loops, and poor performance can be quite difficult to distinguish, and in some cases you need to carry out quite a detailed investigation before deciding which classification is the right one for your problem.

Any of the following symptoms could be caused by a wait, or a loop, or by a badly tuned or overloaded system:

- One or more user tasks in your CICS system fails to start.
- One or more tasks stays suspended.
- One or more tasks fails to complete.
- No output is obtained.
- Terminal activity is reduced, or has ceased.
- The performance of your system is poor.

Because it can be difficult to make a correct classification, consider the evidence carefully before adopting a problem solving strategy. This section gives you guidance about choosing the best classification.

Waits

For the purpose of problem determination, a wait state is regarded as a state in which the execution of a task has been suspended. That is, the task has started to run, but it has been suspended without completing and has subsequently failed to resume.

The task might typically be waiting for a resource that is unavailable, or it might be waiting for an event control block (ECB) to be posted. A wait might affect just a single task, or a group of related tasks. If none of the tasks in a CICS region is running, CICS is in a wait state. The way to handle that situation is dealt with in “What to do if CICS has stalled” on page 124.

If you are authorized to use the CEMT transaction, you can find out which user tasks or CICS-supplied transactions are currently suspended in a running CICS system using the CEMT INQUIRE TASK transaction. Use the transaction several times, perhaps repeating the sequence after a few minutes, to see if any task stays suspended. If you do find such a task, look at the resource type that it is waiting on (the value shown for the HTYPE option). Is it unreasonable that there should be an extended wait on the resource? Does the resource type suggest possible causes of the problem?

You can use EXEC CICS INQUIRE TASK or EXEC CICS INQUIRE TASK LIST commands as alternatives to the CEMT transaction. You can execute these commands under CECI, or from a user-written program. Use the EXEC CICS INQUIRE TASK LIST command to find the task numbers of all SUSPENDED, READY, and RUNNING user tasks. If you use this command repeatedly, you can see which tasks stay suspended. You may also be able to find some relationship between several suspended tasks, perhaps indicating the cause of the wait.

If it seems fairly certain that your problem is correctly classified as a wait, and the cause is not yet apparent, turn to Chapter 6, “Dealing with waits” on page 51 for guidance about solving the problem. However, you should allow for the possibility that a task may stay suspended because of an underlying performance problem, or because some other task may be looping.

If you can find no evidence that a task is waiting for a specific resource, you should not regard this as a wait problem. Consider instead whether it is a loop or a performance problem.

Loops

A loop is the repeated execution of some code. If you have not planned the loop, or if you have designed it into your application but for some reason it fails to terminate, you get a set of symptoms that vary depending on what the code is doing. In some cases, a loop may at first be diagnosed as a wait or a performance problem, because the looping task competes for system resources with other tasks that are not involved in the loop.

The following are some characteristic symptoms of loops:

- The ‘system busy’ symbol is permanently displayed in the operator information area of a display unit, or stays displayed for long periods.
- The transaction abends with abend code AICA.
- CPU usage is very high, perhaps approaching 100%, yet some tasks stay suspended or ready, but not running, for a long time.

  You can check what the CPU usage is for any VSE job by using the DISPLAY SYSTEM ACTIVITY screen of the VSE/ESA Interactive Interface. See the *VSE/ESA Operation* manual for more information.
- There is reduced activity at terminals, or possibly no activity at all.
- One or more CICS partitions appear to be stalled, or to be continuing only slowly.
- No CICS messages are written to any destination, when they are expected.
- No new tasks can be started.
- Existing tasks remain suspended.
- The CEMT transaction cannot be used.
- Repetitive output is obtained. Try looking in these areas:
  - Terminals, and the system console.
  - Temporary storage queues. You can use CEBR to browse them online.
  - Data files and CICS journals.
  - Trace tables, but remember that some loops are intentional—some CICS system tasks use them, for example, to see if there is any work to be done.
- Excessive demand for storage. If the loop contains a GETMAIN request, storage is acquired each time this point in the loop is passed, as long as sufficient storage to satisfy the request remains available. If storage is not also freed in the loop, CICS eventually goes short on storage (SOS) in one of the DSAs. You then get a message reporting that CICS is under stress in one of these areas.

  One further effect is that tasks issuing unconditional GETMAIN requests are suspended more often as the loop continues and storage is progressively used up. Tasks making storage requests do not need to be in the loop to be affected in this way.
- Statistics show a large number of automatically initiated tasks.
- Large numbers of file accesses are shown for an individual task.

Some loops can be made to give some sort of repetitive output. Waits and performance problems never give repetitive output. If the loop produces no output, a repeating pattern can sometimes be obtained by using trace. A procedure for doing this is described in Chapter 7, “Dealing with loops” on page 129.

If you are able to use the CEMT transaction, try issuing the CEMT INQUIRE TASK transaction repeatedly. If the same transaction is shown to be running each time, this is a further indication that the task is looping. However, note that the CEMT transaction itself is always running when you use it to inquire on tasks.

If different transactions are seen to be running, this could still indicate a loop, but one that involves more than just a single transaction.

If you are unable to use the CEMT transaction, it may be because a task is looping and not allowing CICS to regain control. A procedure for investigating this type of situation is described in “What to do if CICS has stalled” on page 124.

Consider the evidence you have so far. Does it indicate a loop? If so, turn to Chapter 7, “Dealing with loops” on page 129, where there are procedures for defining the limits of the loop.

Poor performance

A performance problem is considered to be one in which system performance is perceptibly degraded, either because tasks fail to start running at all, or because they take a long time to complete once they have started.

In extreme cases, some low-priority tasks may be attached but then fail to be dispatched, or some tasks may be suspended and fail to resume. The problem might initially be regarded as a wait.

If you get messages telling you that CICS is under stress, this can indicate either that the system is operating near its maximum capacity, or that a task in error has used up a large amount of storage—possibly because it is looping.

You see one of the following messages when CICS is under stress in one of the DSAs:

DFHSM0131 *applid* **CICS is under stress (short on storage below 16MB)**

DFHSM0133 *applid* **CICS is under stress (short on storage above 16MB)**

If there is no such indication, see Chapter 8, “Dealing with performance problems” on page 139 for advice on investigating the problem. However, before doing so, be as sure as you can that this is best classified as a performance problem, rather than a wait or a loop.

Poor application design

If you have only a poorly defined set of symptoms that might indicate a loop, a wait, or possibly a performance problem with an individual transaction, consider the possibility that poor design might be to blame.

This book does not deal with the principles of application design, or how to check whether poor design is responsible for a problem. However, one example is given here, to show how poor design of an application gave rise to symptoms which were at first thought to indicate a loop.

Environment: CICS and DL/I using secondary indexes. The programmer had made changes to the application to provide better function.

Symptoms: The transaction ran and completed successfully, but response was erratic and seemed to deteriorate as the month passed. Towards the end of the month, the transaction was suspected of looping and was canceled. No other evidence of looping could be found, except that statistics showed a high number of I/Os.

Explanation: The programmer had modified the program to allow the user to compare on the last name of a record instead of the personnel number, which it had done in the past. The database was the type that grew through the month as activity was processed against it.

It was discovered that in making the change, the program was no longer comparing on a field that was part of the key for the secondary index. This meant that instead of searching the index for the key and then going directly for the record, every record in the file had to be read and the field compared. The structure of the source program had not changed significantly; the number of database calls from the program was the same, but the number of I/Os grew from a few to many thousands at the end of the month.

Note that these symptoms eight equally well have pointed to a performance problem, although performance problems are usually due to poorly tuned or overloaded systems, and affect more than just one transaction. Performance problems tend to have system-wide effects.

Where to look next

- For **transaction abends**, see Chapter 4, “Dealing with transaction abends” on page 27
- For **system abends**, see Chapter 5, “Dealing with CICS system abends” on page 39
- For **waits**, see Chapter 6, “Dealing with waits” on page 51
- For **loops**, see Chapter 7, “Dealing with loops” on page 129
- For **poor performance**, see Chapter 8, “Dealing with performance problems” on page 139
- For **incorrect output**, see Chapter 9, “Dealing with incorrect output” on page 149
- For **storage violations**, see Chapter 10, “Dealing with storage violations” on page 173
- For **XRF errors**, see Chapter 11, “Dealing with XRF errors” on page 185

If you have already decided that you should refer the problem to the IBM Support Center, you can find advice about dealing with the Center in Chapter 17, “IBM program support” on page 259.

Chapter 3. Sources of information

This chapter describes the information sources which may be useful in problem determination

User documentation

Have available the collection of information produced by your organization about what your system and applications do and how they do it. It could include:

- Program descriptions or functional specifications
- Record layouts and file descriptions
- Flowcharts
- Statement of inputs and outputs
- Change history of a program
- Change history of your installation
- Auxiliary trace profile for your transaction
- Statistical and monitoring profile showing average inputs, outputs, and response times

Manuals

You should have available the manuals in the CICS Transaction Server for VSE/ESA Release 1 library and the libraries for any other products you use with your application.

Make sure that the level of any manual you refer to matches the level of the system you are using. Problems often arise through using either obsolete information, or information about a level of the product that is not yet installed.

Source listings and link-edit maps

Include the source listings of any applications written at your installation. (They often form the largest single element of documentation. Large installations with thousands of programs might prefer to keep such listings on microfiche or CD-ROM.) Make sure you include the relevant linkage-editor output with your source listings to avoid wasting time trying to find your way through the phases with an out-of-date link map. Be sure to include the JCL at the beginning of your listings, to show the libraries that were used and the sublibrary in which the phase was placed.

Abend codes and error messages

When a CICS transaction stops processing because of an error, the Online Problem Determination (OLPD) program becomes active. OLPD analyzes the data it collects and, if possible, describes the cause of the error. For more information about transaction abends and OLPD, see the *VSE/ESA Guide for Solving Problems*.

© Copyright IBM Corp. 1979, 1999

Messages are sent to several transient data destinations, for example:

- CSMT for terminal error and abend messages
- CSNE for messages issued by DFHZNAC
- CSTL for terminal I/O error messages
- CSFL for file control messages
- CXRF for terminal control messages from an XRF alternate CICS system

For a list of the destinations used by CICS, see the *CICS System Definition Guide*. Use the *VSE/ESA Messages and Codes Volume 3* manual to look up any messages whose meaning you do not know. Make sure that you also have some documentation of application messages and codes for programs that were written at your installation, as well as a copy of the *VSE/ESA Messages and Codes Volume 3* manual.

Symptom strings

CICS produces symptom strings in CICS system and transaction dumps, and in message DFHME0116.

The symptom string provides a number of keywords that can be used to search the RETAIN database. If your installation has access to the IBM INFORMATION/ACCESS licensed program, 5665-266, you can search the RETAIN database yourself. If you report a problem to the IBM Support Center, you are often asked to quote the symptom string.

Although the symptom string is designed to provide keywords for searching the RETAIN database, it can also give you significant information about what was happening at the time the error occurred, and it might suggest an obvious cause or a likely area in which to start your investigation.

Change log

The information in the change log can identify changes made in the data processing environment that may have caused problems with your application program. To make your change log most useful, include the data concerning hardware changes, system software (such as VSE and CICS) changes, application changes, and any modifications made to operating procedures.

Dumps

Dumps are an important source of detailed information about problems. Whether they are the result of an abend or a user request, they allow you to see a snapshot of what was happening in CICS at the moment the dump is taken. Chapter 15, “Using dumps in problem determination” on page 225 contains guidance about using dumps. However, because they do only provide a “snapshot,” you may need to use them in conjunction with other sources of information relating to a longer period of time, such as logs, traces, and statistics.

Statistics

Statistics are often overlooked as a source of debugging information, but those that relate to an application program can help solve problems. It is useful to have a statistical profile (as mentioned in “User documentation” on page 19) to use for problem determination. If you compare the information in the profile with the statistical information produced by CICS, any differences you find may indicate the source of a problem.

Statistics are most often used in system tuning and diagnosis, but they also contain information that can indicate problems with the way your application handles resources. For example, you may notice from these statistics that tables are being loaded, or programs linked, for which there is no known requirement.

You can also use statistics to check terminals, files, queues, and so on for irregularities in their activity. For example, if a terminal has the same number of errors recorded for a particular transaction as the number of times that transaction was run, this may indicate that an incorrect data stream is being sent to that terminal. See the *CICS Performance Guide* for more information about using statistics.

Monitoring

You can use CICS monitoring to provide information for debugging applications. In addition to the system-defined event monitoring points (EMPs) that already exist within CICS code itself, you can define user event monitoring points in your own application programs by using the EXEC CICS MONITOR command.

At a user EMP, you can add your own data (up to 256 counters, up to 256 clocks, and a single character string of up to 8192 bytes) to fields reserved for you in performance class monitoring data records. You can use these extra EMPs to count how many times a certain event happens, or to time the interval between two events. Your definitions in the Monitoring Control Table (MCT) specify the type and number of fields that are available for your use within each task’s performance record. For further information on the MCT see the *CICS Resource Definition Guide*. See the *CICS Application Programming Reference* manual for programming information on syntax and options of the EXEC CICS MONITOR command.

When your monitoring data has been collected, you usually need to format it, which you can do by using a program based on the sample program, DFH$MOLS, or to read it into a database, so that you can analyze it selectively. If you have a suitable MVS™ system, you can transfer your CICS for VSE/ESA monitoring information to it, and then process it using the Service Level Reporter Version 2 (SLR II).

See the *CICS Performance Guide* for guidance about using monitoring data.

Transaction inputs and outputs

Transaction inputs and outputs can be divided into the following areas:

- Terminal data
- Transient data and temporary storage
- Passed information
- Files and databases

Terminal data

Terminal data is very important in solving problems, because it can help you answer the following questions:

- What data did you enter just before the transaction failed?
- Was there any output? If so, what did it look like?

The more you know about the information that was entered at the terminal on which the transaction failed, the better your chance of duplicating the problem in a test environment. However, this information may not be precise, especially if there are many fields on the input screen. You are recommended to provide a quick and easy way for terminal operators to report problems, so that they can report the error while they can still see the data on the screen (or at least remember more clearly what it was).

The questions to consider are:

- Were all necessary input fields entered?
- Were the contents of the input fields correct?
- Which transmit key was used, (that is ENTER, a PF key, or a PA key)?

The output from a transaction is sometimes easier to capture. If you have a locally attached printer, you can make a copy. (The problem may be that the printer output is incorrect.)

On the output screen, check the following points:

- Do all the required fields contain data?
- Is the data correct?
- Is the screen format as designed?
- Are there any nondisplay fields used to pass data that may not be protected?

Transient data and temporary storage

If the program explicitly uses any transient data or temporary storage queues, inspect them to see if their content is as expected. You can use the CICS-supplied transaction, CEBR, to inspect temporary storage queues in some detail. See the *CICS-Supplied Transactions* manual for information about this transaction.

Even if the program does not use queues, look at the system queues for CEMT (or your site replacement) and CSTL to see if there are any relevant messages.

The things you might want to look for in the queues are:

- Are the required entries there?
- Are the entries in the correct order?
- Is the queue being written the same one that is being read?

Passed information

Whenever you suspect a problem, check any information that is passed in any of these ways:

- COMMAREA
- Transaction work area (TWA)
- GETMAINed area, pointer in TWA
- In-memory table
- Common system area (CSA) work area
- Terminal user area

Be particularly careful when you are using the common work area (CWA) because you only have one area for the entire system. A transaction may depend on a certain sequence of transactions and some other program may change that sequence. If you are using the CWA, you must also know if your CICS is split into multiple multiregion operation (MRO) regions because there is an independent CWA for each MRO region.

Terminal user areas can have problems because the area is associated with a terminal and not a particular transaction.

If you are using tables in the CWA, remember that there is no recovery; if a transaction updates the table and then abends, the transaction is backed out but the change is not.

Files and databases

Files and databases are often the main source of transaction input and output—you should always investigate both these areas whenever a program is having problems. To do this, you need to use the appropriate utilities and diagnostic tools for the data access methods that you have at your installation.

Check the various indexes in files and databases. If you have more than one method of accessing information, one path may be working well but another path may be causing problems.

When looking through the data in files, pay particular attention to the record layout. The program may be using an out-of-date record description.

Traces

CICS provides a tracing facility that enables you to trace transactions through the CICS components as well as through your own programs. CICS auxiliary trace enables you to write trace records on a sequential device for later analysis. For information about the tracing facilities provided by CICS, read Chapter 14, “Using traces in problem determination” on page 199.

VSE service aids

The VSE service aids that are most likely to be of use in solving CICS problems are:

- System debugging aid (SDAID), which traces instructions, branches or storage or register alterations.

- EREP (Environmental Record Editing and Printing Program), which reads error and other environmental records generated by hardware and software, edits the records, and produces printed reports at your request.

See the *VSE/ESA Guide to Solving Problems* for further information on SDAID and EREP.

Part 2. Dealing with the problem

Part 2 contains:

Chapter 4. Dealing with transaction abends 27
Collecting the evidence 27
What the abend code can tell you 28
CICS transaction abend codes 28
Finding where a program check occurred 30
Analyzing the problem further 36
FEPI abends 38

Chapter 5. Dealing with CICS system abends 39
The documentation you need 39
Interpreting the evidence 40

Chapter 6. Dealing with waits 51
Techniques for investigating waits 52
How tasks are made to wait 59
Spooler waits 77
Transaction manager waits 78
Lock manager waits 82
Journal control waits 85
Task control waits 88
Storage waits 90
Temporary storage waits 91
Terminal waits 94
VTAM terminal control waits 102
Interregion and intersystem communication waits 104
Transient data waits 105
Loader waits 108
File control waits 108
Interval control waits 115
XRF alternate system waits 121
CICS system task waits 122
FEPI waits 123
What to do if CICS has stalled 124

Chapter 7. Dealing with loops 129
What sort of loop is indicated by the symptoms? 129
Investigating loops that cause transactions to abend with abend code AICA 134
Investigating loops that are not detected by CICS 136
What to do if you cannot find the reason for a loop 138

Chapter 8. Dealing with performance problems 139
Finding the bottleneck 140
Finding why tasks fail to get attached to the transaction manager 141
Finding why tasks fail to get attached to the dispatcher 142
Finding why tasks fail to get an initial dispatch 144
Finding out why tasks take a long time to complete 146
A summary of performance bottlenecks, symptoms, and causes 146

© Copyright IBM Corp. 1979, 1999

Chapter 9. Dealing with incorrect output 149
Trace output is incorrect 149
Dump output is incorrect 153
Wrong data has been displayed on a terminal 156
Incorrect data is present on a VSAM data set 163
An application did not work as expected 163
Your transaction produced no output at all 164
Your transaction produced some output, but it was wrong 169

Chapter 10. Dealing with storage violations 173
Avoiding storage violations 173
Two kinds of storage violation 174
CICS has detected a storage violation 174
Storage violations that affect innocent transactions 181
Programming errors that can cause storage violations 182
Storage recovery 183

Chapter 11. Dealing with XRF errors 185
Symptoms of problems in an XRF complex 185
Debugging the overseer sample program 189
XRF trace 190
The CAVM data sets 190

Chapter 12. External CICS interface 193

Chapter 13. Dealing with MRO problems 195

Chapter 4. Dealing with transaction abends

This chapter gives guidance about finding the cause of transaction abends. It contains the following topics:

- "Collecting the evidence"
- "What the abend code can tell you" on page 28
- "CICS transaction abend codes" on page 28
- "Finding where a program check occurred" on page 30
- "Analyzing the problem further" on page 36
- "FEPI abends" on page 38

When a CICS transaction abends (ends abnormally), a transaction abend message and an abend code of four alphanumeric characters are sent to CSMT, the CEMT transient data destination (or your site replacement). This is an example of what the message looks like:

DFHAC2006 *date time applid* **Transaction** *tranid* **program** *program name* **abend** *primary abcode* **at** *termid.*

The message contains several vital pieces of information. It identifies the transaction (*tranid*) that failed, and the program (*program name*) that was being executed when the failure was detected. Most importantly, it gives you the abend code (*abcode*), indicating the nature of the error.

The transaction abend can originate from several places, and the method you use for problem determination depends on the source of the abend. The procedures are described in the sections that follow. As you go through them, you might like to use the worksheet that is included at the end of this chapter to record your findings ("Worksheet for transaction abends" on page 37).

Collecting the evidence

You can find all the evidence needed to investigate a transaction abend in the information sent to the various transient data queues for error messages, and in the transaction dump. If no transaction dump has been produced, it is possible that transaction dumping has been suppressed for the transaction (via the transaction definition), or the dump code entry in the transaction dump code table suppresses dumping. For guidance about changing the dumping options so that you get a transaction dump, see Chapter 15, "Using dumps in problem determination" on page 225.

The transaction abend code and the abend message are recorded in the CSMT log. Make a note, too, of any other messages you find there that might relate to the abend, as they could provide additional valuable evidence.

Check also to see if any relevant messages have been sent to the transient data destinations used by CICS to record messages. For a list of such destinations, see the *CICS System Definition Guide*. Look in particular for any messages about files, terminals, or printers that you might be attempting to use.

© Copyright IBM Corp. 1979, 1999

Symptom string

CICS produces a symptom string as part of the transaction dump. The symptom string gives some details about the circumstances of the transaction dump. It might show, for example, that the dump was taken because the transaction abended with the abend code ASRA.

If you refer the problem that caused the dump to be taken to the IBM Support Center, they will use the symptom string to search the RETAIN database for problems similar to it. For an introduction to symptom strings and their contents, see “Looking at the symptom string in the dump” on page 41.

What the abend code can tell you

The first thing that the abend code can indicate is whether or not this was a CICS abend. CICS transaction abend codes begin with the letter “A”. A user program or another product might also use abend codes beginning with “A”. However, if the transaction abend code begins with anything other than “A”, it is an abend code belonging to a user program or to some other product. For the sake of convenience, all such non-CICS abend codes are referred to in this section as user abend codes.

For an introduction to the types of transaction abend codes used by CICS and by other IBM products, see the *VSE/ESA Messages and Codes Volume 3* manual.

If you have received a user abend code, it can still be difficult to find out which program is responsible for it unless you have adequate documentation. For this reason, it is good practice for all programmers who issue abends from within their programs to document the codes in a central location at your installation.

As far as vendor products are concerned, the documentation includes, in most cases, a list of abend codes that are issued from the programs making up the products. This list, together with the documentation for your internal applications, should make it possible for you to find what caused the abend. If it is not clear why the user abend was issued, you might need to describe the problem to the owner of the program.

CICS transaction abend codes

The best source of information on CICS abends is the *VSE/ESA Messages and Codes Volume 3* manual. That book contains a section that lists all transaction abend codes issued by CICS. There is an explanation of why the code was issued, followed by details of system and user actions.

If, after reviewing the material in the *VSE/ESA Messages and Codes Volume 3* manual, you cannot find the cause of the problem, continue with the procedures described here. The abend codes AICA, ASRA, ASRB, and ASRD are dealt with separately because special procedures apply to them. If your abend code was something other than these, use the procedures in “Locating the last command or statement” on page 241, to find the last command that was executed, and then turn to “Analyzing the problem further” on page 36.

AICA abends

Abend code AICA normally indicates that the transaction has been in a loop. You can find detailed guidance about this subject in Chapter 7, “Dealing with loops” on page 129.

ASRA abends

CICS issues an ASRA abend code when it detects that a program check has occurred within a transaction. Program checks can occur for a wide variety of reasons, but you can find the nature of the error from the program interrupt code in the program status word (PSW). The PSW is used by the machine hardware to record the address of the current instruction being executed, the addressing mode, and other control information. The PSW gives you the address at which the program check occurred, and so it represents a record of the circumstances of the failure. See “Finding where a program check occurred” on page 30 for more information.

ASRB abends

A transaction can abend with an abend code of ASRB when a program issues the VSE ABEND macro. For example, DAM issues this ABEND macro when it detects errors, rather than sending a return code to the calling program. CICS is notified when a VSE abend occurs, and in turn issues an ASRB abend code for the transaction.

Use the procedures outlined in “Locating the last command or statement” on page 241 to find the origin of the abend in your program. That information, together with the description and procedures for ASRB abends given in the *VSE/ESA Messages and Codes Volume 3* manual, should be sufficient for you to solve the problem.

ASRD abends

A transaction abends with code ASRD if:

- An application program attempts to invoke CICS macros.
- An application program attempts to access the CSA or TCA.
- An application program issues an EXEC CICS ADDRESS CSA command, and attempts to access storage addressed by the pointer that is returned.
- A COBOL application program attempts to access the CSA via a BLL cell.

Any of the above causes a program check that CICS diagnoses as an ASRD abend, rather than the usual ASRA abend. You can use the information in the PSW to investigate the cause of an ASRD abend.

AEYD abends

If command protection is activated by the CMDPROT(YES) system initialization parameter the AEYD transaction abend can occur. CICS terminates a transaction with this code when an output parameter of an EXEC CICS command addresses storage that the issuing transaction could not itself directly overwrite.

At the time of the abend, register 2 points to the parameter area containing the invalid address. The trace should include an exception trace entry created by DFHEISR. This entry should identify the parameter in error. If the abend is

handled, EXEC CICS ASSIGN ASRASTG, ASRAKEY, and ASRAREGS commands can give additional information.

To prevent a recurrence of the abend, it is recommended that you correct the program code. Alternatively, changing one or more of the following options may alleviate the problem:

- EXECKEY in the RDO PROGRAM definition, if storage protection is active
- TASKDATAKEY in the RDO TRANSACTION definition

For further information, see “Avoiding storage violations” on page 173.

Finding where a program check occurred

When a transaction abends with code ASRA or ASRD, the first thing you need to do is find out where the program check occurred. CICS will have attempted to establish this for you. A record of the program in error and the offset of the program check within the program load module are contained in the following places:

- Message DFHAP0001 or DFHSR0001, which precedes an abend
- The transaction abend control block (TACB) which is created to describe an abend
- Exception trace point ID AP 0781 (for an ASRA abend) or AP 0783 (for an ASRD abend)

See “Interpreting transaction dumps” on page 237 for details of the contents of a transaction dump, including the TACB.

The offset indicates the point in the program at which the program check occurred. Note that the offset is derived from the PSW next sequential instruction address and so may indicate the instruction after the one that failed. Unless the offset is X'FFFFFFFF', turn to “What type of program check occurred?” on page 31.

If the offset appears as X'FFFFFFFF', CICS was unable to establish the location of the program check. If this is the case, use the PSW to obtain the next sequential instruction address. The PSW may be found in the following places:

- The TACB for the abend
- At the head of the formatted transaction dump
- Within the kernel error data block traced by exception trace point IDs AP 0781 or AP 0783

Now note down the start and end addresses of the different program areas in the transaction dump. Is the next sequential instruction address from the PSW in any of the programs? If so, then that is the program in which the interrupt occurred. Use the procedure described in “Locating the last command or statement” on page 241 to identify the last command executed.

If the address is *outside* all of the programs, one of two things is likely to have happened.

- The program in which the program check occurred was running on your behalf (for example, VSAM or DL/I), but not under CICS control. This is usually

caused by incorrect parameters being passed to the program, or parameters being passed in the wrong sequence. These are usually caught and flagged with an appropriate return code, but certain combinations can cause problems.

- Your program might have taken a wild branch into some other piece of storage. If the address from the PSW ends in an odd number, this is probably the case, as valid instructions are always on an even address. The address could be within the CICS address space, or, indeed, anywhere else in virtual storage. Often, a wild branch is taken to address zero, because the register that should contain the branch address is set to zero. The PSW usually contains address X'00000004' after such a branch has occurred.

Check the register contents to see whether any of them contains the next sequential instruction address from the PSW, or something close to it. This might help you find out how you got to the wrong address.

If the PSW does point to an instruction in one of your programs, the next thing to consider is the type of program check that occurred. Otherwise, turn directly to "Analyzing the problem further" on page 36.

What type of program check occurred?

Knowing what type of program check occurred can be helpful in finding the cause of the error. This is indicated by the program interrupt code (PIC), which you can find in the PSW at the start of the transaction dump. You can find information about the PSW in the *ESA/390 Principles of Operation* manual.

PIC	PIC explanation
1	Operation exception—incorrect operation attempted. Possible causes: • Overlaid program • Overlaid register save area, causing incorrect branch • Resource unavailable, but program logic assumed valid address was returned, and took inappropriate action • Incorrect branch to data that contains no instruction known to the machine • In an assembler-language program, a base register was inadvertently changed
2	Privileged operation—this program is not authorized to execute this instruction Possible causes: • Incorrect branch to this code; may be due to: – Overlaid register save area – Program overlaid by data that contains the privileged operation code
3	Execution exception—you are not allowed to EXECUTE an EXECUTE instruction. Possible causes: • Incorrect branch to this code

- Incorrect register contents, may be due to:
  - Overlaid register save area
  - Program overlaid by data that contains the incorrect instruction
  - Incorrect program logic

4 Protection exception—read or write access violation has occurred.

Possible causes:

- Resource unavailable, and return code not checked. Program logic assumed valid address returned and took inappropriate action.
- Incorrect interface parameters to some other program or subsystem (for example, VSAM or DL/I).
- Overlaid register save area, causing incorrect reference to data.
- In an assembler-language program, incorrect initialization or modification of a register used to address data.
- Attempt to access internal control blocks illegally or used a CICS system or application programming macro call.
- Attempt to write to storage for which the application does not have an adequate key. For example, in a CICS system with storage protection, an application running in USER key attempts to write to the CDSA, the RDSA, the ECDSA, or the ERDSA.
- Attempt to write to the ERDSA or RDSA when PROTECT is specified for the RENTPGM system initialization parameter.
- Storage, passed to CICS as an output parameter through the EXEC interface, that is not addressable by the application issuing the call. The transaction is abended AEYD, and the PSW shows that a protection exception has occurred.

5 Addressing exception—the address that you referenced is not available or is not valid.

Possible cause:

- Incorrect register contents, may be due to an overlaid register save area

6 Specification exception—incorrect format of an instruction or invalid registers.

Possible causes:

- Overlaid program
- Incorrect field lengths used in packed-decimal multiply and divide instructions
- Branch to an odd-numbered address, caused by an overlaid register save area

7 Data exception—data invalid in a packed or signed-display decimal operation. One, or possibly both, of the operands contain data not suitable for the instruction.

Possible causes:

- Incorrect input data (often because blanks have been used where numeric data is expected)
- Overlaid data
- Overlaid register save area, causing an incorrect branch
- Incorrect program logic, execution of code with uninitialized variables
- Wrong length

8 through F Arithmetic exceptions, such as divide checks, overflow, and underflow. They differ in the form of arithmetic that was being used—binary, packed decimal, or floating point.

Possible causes:

- Incorrect user data
- Overlaid data areas
- Overlaid register save area, causing incorrect reference to data

10 and above Program checks associated with system-related interrupts.

Dealing with arithmetic exceptions

If the program check was due to an arithmetic error (interruption codes 8 through F), you need to find the operands used in the last instruction. Use the procedure described in “Locating program data” on page 242 to locate the fields. You need to know a little about the type of arithmetic being done, so that you can tell if the operands are valid. The interrupt you received tells you what sort of arithmetic the system was doing (binary, packed decimal, or floating point), but you need to determine if that is what you had intended to do. You might need to consult a programming language manual if you have any queries about this.

When you have identified the operands, you need to decide where the problem is. Questions to consider include:

- Has the data been overlaid?
- Has the value been changed by faulty logic?
- Does the data type not match the operation type? For example, if you define the variable as being packed decimal and then you read in binary information, this causes a ‘data exception’ error.

Dealing with protection exceptions

With the storage protection facility, there are further situations in which a protection exception (interrupt code 4) may occur:

- An attempt is made to write to the CDSA, RDSA, ECDSA, or ERDSA, when storage protection is active and the application is running in user key
- An attempt is made to write to the ERDSA or RDSA when PROTECT is specified for the RENTPGM system initialization parameter.

If any of these events occurs, CICS abnormally terminates the transaction with abend code ASRA and issues message DFHSR0622 which identifies the DSA over which the program attempted to write. This information is in the TACB and is traced by exception trace point ID AP 0781. It is also useful to know the execution

key of the program at the time of the protection exception. This appears in the TACB, exception trace point ID AP 0781 and at the head of the formatted transaction dump.

If the command protection facility is enabled (by specification of CMDPROT=YES in the system initialization parameter), a protection exception can occur if storage, passed to CICS as an output parameter through the EXEC interface, is not accessible for READ/WRITE by the program issuing the command. The program is passing to CICS storage that it cannot itself update, but it requires CICS to do so for it. The transaction terminates abnormally with abend code AEYD. CICS creates an exception trace entry AP 0779 and saves relevant data in the TACB that is formatted at the beginning of the transaction dump.

Note

Storage protection, and command protection are facilities that add data integrity by highlighting application errors. In previous releases of CICS, such errors may not have been detected or may have appeared as CICS problems. The use of these facilities greatly reduces the number of abends that appear to be CICS problems.

It is still possible for CICS to abend when the problem is in the application. For example, command protection only checks output parameters and does not prevent the passing of fetch-protected storage as an input parameter to CICS. When CICS attempts to read such storage, an ASRA abend occurs.

Causes of protection exceptions

CICS storage protection is intended to prevent application programs erroneously overwriting CICS programs and control blocks. The occurrence of a protection exception in a new program running in a system with storage protection active probably indicates an error in the application program. However, when existing programs which need to be defined with EXECKEY(CICS) are first migrated to a system with storage protection active, protection exceptions may well occur.

Any application program causing a protection exception when defined with EXECKEY(USER) must be examined to determine why it is attempting to modify storage that it is not allowed to modify. Its definition should be changed to EXECKEY(CICS) only if it is determined that the application program is legitimately accessing CICS key storage, and the exception is not the result of an application error.

Programs might also be incorrectly link-edited as reentrant (SVA option on PHASE linkage editor statement) and, as a result, loaded by CICS into one of the read-only DSAs (RDSA or ERDSA). When such an incorrectly defined program attempts to modify itself, or another program tries to modify it, a protection exception occurs. The program should be checked to see whether it should be redefined as non-reentrant, or whether the program should be changed to be truly reentrant. The protection exception might indicate that the program uses poor programming techniques that could result in other problems if uncorrected.

Command protection

Command protection (activated by specifying CMDPROT=YES in the system initialization parameter) prevents CICS from updating storage if the storage address is passed as a command output parameter by a transaction that is not authorized to update that storage. The transaction terminates with abend code AEYD. The exception trace entry AP 0779 supplies details of the failing program and command. When migrating to a system with command protection enabled, EXEC commands that pass unauthorized storage are identified and can be corrected.

Possible causes of protection exceptions referencing CICS DSAs

The following is a summary of some of the causes of protection exceptions that can occur in user key programs:

- Issuing a VSE macro request. Most VSE macros and services are not supported in EXECKEY(USER) application programs. Use of unsupported macros and services may cause a failure if these macros or services attempt to reference VSE storage outside the CICS DSAs.
- Referencing storage obtained by an VSE GETVIS request or another VSE macro. VSE storage obtained by these methods resides outside the CICS DSAs, and is therefore protected from user key programs.
- Using PL/I statements, COBOL verbs or compiler options that are not permitted in CICS application programs (see the *CICS Application Programming Guide* for details of prohibited language statements and compiler options). For example, the use of CALL with the RES compiler option, or a verb such as INSPECT, may also cause VSE storage outside the CICS DSAs to be obtained or updated (such storage is protected from user-key programs).

  In previous versions of CICS, these may have worked, or at least may not have caused the application to fail. However, the use of these statements and options can have other effects on the overall execution of the CICS system, and should be removed where possible.
- Modifying the CWA when CWAKEY=CICS is specified as a system initialization parameter. In a user key program, this is an invalid reference to storage allocated from the CDSA or ECDSA.
- Modifying the TCTUA when TCTUAKEY=CICS is specified as a system initialization parameter. In a user key program this is an invalid reference to storage allocated from the CDSA or ECDSA.
- Issuing an EXEC CICS EXTRACT EXIT command and attempting to update an exit program's global work area. In a user key program this is an invalid reference to storage allocated from the CDSA or ECDSA.

Protection exceptions referencing the read-only DSAs

Protection exceptions occurring in programs resident in the ERDSA and RDSA are caused by the program not being truly reentrant. It might be that the program should not be defined as reentrant, or it might be that the program should be reentrant but is using poor coding techniques which should be corrected instead. For example:

- Using static variables or constants for fields which are set by CICS requests. For example, in assembler coding, if the LENGTH parameter for a retrieval operation such as EXEC CICS READQ TS is specified as a DC elsewhere in the program, a constant is set up in static storage. When CICS attempts to set

the actual length into the data area, it causes a protection exception if the program is in the ERDSA or RDSA.

In some cases, for example EXEC CICS READ DATASET INTO () LENGTH() ..., the LENGTH value specifies the maximum length that the application can accept, and is set by CICS to contain the actual length read on completion of the operation. Even if the program does not have SVA specified on the linkage-editor PHASE statement, using a static variable itself for this length could cause problems if the program is being executed concurrently for multiple users. The first transaction may execute correctly, resulting in the actual record length being set in the LENGTH parameter, which is then used as the maximum length for the second transaction.

- Defining a table with the SVA linkage-editor option on the PHASE linkage editor statement and then attempting to initialize or update the table during CICS execution. Such a table should not be defined with the SVA option.
- Defining BMS mapsets as with the SVA linkage-editor option can cause a protection exception, if CICS attempts to modify the mapsets. In some cases, CICS needs to modify BMS mapsets during execution. Mapsets should not be link-edited with the SVA option. BMS mapsets should be loaded into CICS key storage (because they should not be modified by application programs) which means they must not be link-edited with the SVA option. (Partition sets are not modified by CICS and can be link-edited with the SVA option.)

See the *VSE/ESA System Control Statements* manual for further information on the PHASE linkage editor statement and the SVA option.

Analyzing the problem further

You should now know the point in the program at which the abend occurred, and what the program was attempting to do.

- If your program uses or calls other programs or systems, examine the interface and the way you pass data to the program. Are you checking the returned information from the other system? Incorrect logic paths based on incorrect assumptions can give unpredictable results.
- Examine the flow of your program using tools like the Execution Diagnostic Facility (CEDF). Check the transient data and temporary storage queues with the CICS browse transaction (CEBR), and use the CICS command-level interpreter and syntax checker transactions (CECI and CECS). If necessary, insert additional statements into the program until you understand the flow.
- Look at any trace output you might have. If you have a “normal” trace output included in the documentation, compare the two for differences.
- Define the current environment, and try to isolate any changes in it since your program last worked. This can be difficult in large installations, because so many people interact with the systems and slight changes can affect things that seem unconnected.

Worksheet for transaction abends

1. Record the abend code and messages
   Find the abend code from the heading of the dump and record any pertinent messages.

2. Is this a CICS or a USER abend code?
   For a CICS abend code, go to step 3.
   If this is a USER abend code, tell the appropriate person.
3. Look up the abend code
   If you need further advice, go to step 4.
4. Is this an AICA abend?
   Yes; read Chapter 7, “Dealing with Loops”.
   No; go to step 5.
5. Is this an ASRA abend?
   Yes; go to step 7.
   No; go to step 6.
6. Is this an ASRD abend?
   Yes; go to step 7.
   No; go to step 14.
7. Record the program areas from the dump.

   Find the program names from the Module Index at the end of the formatted dump.

Begin Address	End Address	Program Name

8. Record the address of the next instruction from the PSW, or the offset established by CICS.

Offset/address of next instruction	

9. Did the program check occur in one of the program areas listed above?
   Yes; go to step 10.
   No; go to step 14.
10. Record what type of program check occurred.

Program Interrupt Code	

11. Find the last statement executed.

    Last Statement Executed

    (See “Locating the last command or statement“ in chapter 15)
12. Was the PIC one of the arithmetic interrupts
    (7,8,9,A,B,C,D,E,F)?
       No; go to step 15.
       Yes; find the contents of the operands of the last instruction.

    Contents

    (See “Locating program data” in chapter 15)
       Now go to step 15.
13. Was the PIC a protection exception?
       No; go to step 15.
       Yes; see
       and go to step 15.
14. Find the last statement executed.

    Last Statement Executed

    (See “Locating the last command or statement” in chapter 15)
15. Analyze the problem and the data gathered.
    For most problems you should now have enough information to solve the
    problem. If you still cannot find the source, recheck the following:
    1. Parameters to or from other programs or systems.
    2. Any needed resource that may not be available.
    3. The formatted trace, for any unexplained flow.
    4. The running environment, for any changes in it.

FEPI abends

For information about FEPI-associated abends in CICS or VSE, see the *CICS Front End Programming Interface User's Guide*.

Chapter 5. Dealing with CICS system abends

The purpose of this chapter is to tell you how to gather essential information about CICS system abends. You are likely to be reading it for any of these reasons:

- A message has alerted you that a CICS system abend has occurred.
- You have been directed here from the *VSE/ESA Messages and Codes Volume 3* manual, or the symptom code given in a message has prompted you to look here.
- The symptom string at the head of a system dump gives insufficient information about the cause of a CICS system abend

If you have not yet done so, look in the *VSE/ESA Messages and Codes Volume 3* manual for an explanation of any message you may have received, because it could offer a straightforward solution to your problem.

If the abend was clearly caused by a storage violation, turn directly to Chapter 10, “Dealing with storage violations” on page 173. You know when CICS has detected a storage violation, because it issues this message:

DFHSM0102 *applid* **A storage violation (code X'***code***') has been detected by module** *modname*.

On reading this chapter, you may find that the abend was due to an application error. In this case, you need to look at the application to find out why it caused the abend. However, if you find that a CICS module seems to be in error, you need to contact the IBM Support Center. Before doing so, you must gather this information:

- The name of the failing module, and the module level
- The offset within the module at which the failure occurred
- The instruction at that offset
- The abend type

This chapter tells you how to find out all of these things, under:

- “The documentation you need”
- “Interpreting the evidence” on page 40

The documentation you need

The primary documentation you need for investigating system abends is the system dump, taken at the time the error occurred. This usually contains all the evidence needed to find the cause of the problem.

If system dumping is permitted for the dump code, and has not otherwise been disabled, a system dump will have been taken when the error was detected. You can find out which dump relates to which message, because the time stamps and the dump IDs are the same.

If a system dump was not taken when the abend occurred, you need to find out why. Use the procedure described in “You did not get a dump when an abend occurred” on page 153, and follow the advice given there. When you are sure that

© Copyright IBM Corp. 1979, 1999

dumping is enabled for the appropriate system dump code, recreate the system abend.

You can use the system dump formatting program, DFHPD410, to format and print the dump. Details of how to do this are given in the *CICS Operations and Utilities Guide*. The kernel domain storage areas (formatting keyword KE) and the internal trace table (formatting keyword TR) are likely to be the most useful at the start of your investigation.

The formatted output for kernel domain (search for the eye-catcher `===KE`) contains summary information about the error. The internal trace table (eye-catcher `===TR`) contains the exception trace entry (if any) that was made at the time the error was detected.

Later, you might find that storage summaries for the application, transaction manager, program manager, dispatcher, and loader domains (formatting keywords AP, XM, PG, DS, and LD, respectively) are also useful. In each case, level-1 formatting is sufficient in the first instance.

You may need to copy the dump so that you can leave the system dump data set free for use, or so that you have a more permanent copy for problem reporting.

Whether you look at the dump online or offline, do not purge it from the dump data set until you have either copied it or finished with it—you might need to format other areas later, or the same areas in more detail.

Interpreting the evidence

The first things to look at are any messages accompanying the abend, the exception trace entry in the internal trace table, and the symptom string at the start of the dump.

Looking at the messages

Messages that accompany a CICS system abend can sometimes point directly to the cause of the failure. For every case, advice about how to react to a message is given in the *VSE/ESA Messages and Codes Volume 3* manual.

Looking at the exception trace entry

When a CICS system abend occurs, an exception trace entry is made to the internal trace table and any other active trace destination. It does not matter whether you have tracing turned on or not—the trace entry is still made.

If the trace table contains more than one exception trace entry, it is likely that the last one is associated with the dump. However, this might not always be the case, and you should make sure that you have found the correct entry. Be aware, too, that dumps can sometimes be requested without a corresponding exception trace entry being made.

The exception trace entry gives information about what was happening when the failure occurred, and data that was being used at the time.

For details of trace entries, see Chapter 14, “Using traces in problem determination” on page 199.

Looking at the symptom string in the dump

The symptom string in a system dump is similar to the short symptom string at the beginning of a CICS transaction dump. It:

- Is issued as part of message DFHME0116
- Appears at the beginning of a CICS system (VSE SDUMP)

The symptom string provides a number of keywords that can be directly typed into RETAIN and used to search the RETAIN database. The possible keywords are shown in Table 3. The keywords are used at the IBM Support Center to discover duplicate problems, or problems that have already been reported by other users and for which a solution is available. If you have the IBM INFORMATION/ACCESS licensed program, 5665-266, you can search the RETAIN database yourself.

If you report a problem to the IBM Support Center, you are often asked to quote the symptom string.

Table 3. Symptom string keywords

Keyword	**Meaning**
PIDS/	Product ID (CICS product number)
LVLS/	Level indicator (CICS release level)
RIDS/	Module name
PTFS/	Module PTF level
MS/	Message ID reporting error
AB/	Abend code
ADRS/	Address or offset indicator
PRCS/	Return code
PCSS/	CICS jobname
OVS/	Overlaid storage
FLDS/	Name of a field associated with problem
REGS/	Software register associated with problem
VALU/	Value of a named field or register

Although the symptom string is designed to provide keywords for searching the RETAIN database, it can also give you significant information about what was happening at the time the error occurred, and it might suggest an obvious cause or a likely area in which to start your investigation. Amongst other things, it might contain the abend code. If you have not already done so, look in the *VSE/ESA Messages and Codes Volume 3* manual to see what action it suggests for this abend code.

If the system is unable to gather much information about the error, the symptom string is less specific. In such cases, it might not help you much with problem determination, and you need to look at other parts of the dump. The kernel domain storage summary is a good place to start.

Looking at the kernel domain storage areas

The type of information that you can gather from the kernel domain storage areas is as follows:

- A summary of tasks and their status, and whether or not they were in error when the dump was taken.
- An error analysis report for each task currently in error.
- CICS retains information for the previous fifty errors.
- The linkage stack for each task, showing which programs have been called and have not yet returned.

The first thing you need to do is to find out which tasks are associated with the error.

Finding which tasks are associated with the error

You can find out which tasks are associated with the error from the kernel task summary. This tells you which tasks were in the system when the dump was taken, whether or not they were running, and whether they were in error.

The task summary is in the form of a table, each line in the table representing a different task.

The left-hand column of the task summary shows the kernel task number, which is the number used by the kernel domain to identify the task. This is not the same as the normal CICS task number taken from field TCAKCTTA of the TCA.

Figure 1 shows an example of a kernel task summary with a task in error.

```
===KE: KERNEL DOMAIN KE_TASK SUMMARY
KE_NUM KE_TASK  STATUS       TCA_ADDR TRAN_# TRANSID DS_TASK  KE_KTCB  ERROR
0001   0397F930 KTCB STEP    00000000                00000000 039D0300
0002   0397FC20 KTCB QR      00000000                054C9000 039D2100
0003   0397FF10 KTCB RO      00000000                054CA000 039D1200
0004   03980200 NOT RUNNING  0007D5F0 00046  CECI    054D05C8 039D2100
0005   039804F0 NOT RUNNING  000805F0 00055  CEMT    054D0AA8 039D2100
0006   039807E0 UNUSED
0007   03980AD0 NOT RUNNING  03B86000 00058  CEDF    054D0428 039D2100
0008   03980DC0 NOT RUNNING  00077000 00038  ENA1    054D0288 039D2100
0009   039810B0 NOT RUNNING  00074000 00006  CSSY    054D01B8 039D2100
000A   039813A0 NOT RUNNING  0007D000 JBS    CSSY    054D09D8 039D2100
000B   03981690 NOT RUNNING  03B1B5F0 00004  CSNE    054D0908 039D2100
000C   03981980 NOT RUNNING  000775F0 J01    CSSY    054D0768 039D2100
000D   03981C70 NOT RUNNING  000745F0 00021  CSSY    054D0B78 039D2100
000E   03981F60 NOT RUNNING  0006A5F0 00005  CSSY    054D0838 039D2100
000F   03982250 NOT RUNNING  03B1B000 TCP    CSTP    054D0358 039D2100
0010   03982540 ***RUNNING** 00080000 00059  CESN    054D0DE8 039D2100 *YES*
0011   03982830 NOT RUNNING  0006A000 00036  ALL     054D0018 039D2100
0012   03982B20 UNUSED
0013   03982E10 NOT RUNNING  00000000                054D00E8 039D2100
0014   03983100 NOT RUNNING  00000000                054D0D18 039D2100
0015   039833F0 UNUSED
0016   039836E0 UNUSED
0017   039839D0 UNUSED
0018   03983CC0 UNUSED
0019   03983FB0 UNUSED
001A   039842A0 UNUSED
001B   03984590 UNUSED
```

Figure 1. Kernel task summary showing a task in error

Follow these steps to interpret the data in the kernel task summary:

1. When you have located the task summary table in the formatted dump, look in the ERROR column. If you find a value of *YES* for a particular task, that task was in error *at the time the dump was taken*.

   Note: If the recovery routine that is invoked when the error occurs does not request a system dump, you will not see any tasks flagged in error. In such a case, the system dump is likely to have been requested by a program that is being executed lower down the linkage stack and that received an abnormal response following recovery. The program that received the error has gone from the stack, and so cannot be flagged. However, error data for the failing task was captured in the kernel domain error table (see "Finding more information about the error" on page 44). Error data is captured in the error table even when no system dump is taken at all.

   In Figure 1 on page 42, you can see that kernel task number 0010 is shown to be in error.

2. Look next at the STATUS column. For each task you can see one of the following values:

   *****RUNNING****
   The task was running when the system dump was taken. This is the normal status of a task in error.

   NOT RUNNING
   The task is in the system but is currently not running. It may, for example, be suspended because it is waiting for some resource, or it may be ready to run but waiting

   Tasks shown to be "Not Running" are less likely to be associated with the error, but it is possible that one of these could have been flagged with an error. If you find this to be so, the most likely explanation is that the task in error was attempting recovery when, for some reason, it was suspended. to be dispatched.

   KTCB
   The CICS control blocks corresponding to the VSE subtasks. These are treated as tasks in the kernel task summary.

   Unused
   Either the task was in the system but it has now terminated, or there has not yet been a task in the system with the corresponding task number. Earlier "Unused" tasks are likely to have run and terminated, and later ones are likely never to have represented actual tasks. It is most unlikely that you will ever need to distinguish between the two possibilities.

3. Look at the TRAN_# and KE_NUM columns. These are particularly important in solving problems that require the use of traces. The TRAN_# column for a task can contain:

   - A number that matches the task number in the corresponding trace
   - "TCP" for the CICS terminal control task
   - Other character entries for CICS system tasks (for example, a component identifier like "AP" for a CICS system task in the AP domain)

   When you are working with trace output, you can use the number from the TRAN_# column to identify entries associated with a user task up to the point at which that task passes control to CICS. To identify the CICS processing

associated with the user task, you need to use the entry in the KE_NUM column of the kernel task summary. This matches the KE_NUM shown in the full trace entries for the task, and enables you to distinguish the CICS processing associated with the task you are interested in from other CICS processing.

Finding more information about the error

More information about the failure is given in the summary information for the task in error (after the kernel task summary). It gives you a storage report for the task, including registers and PSWs (program status word), and any data addressed by the registers. The PSW is used by the machine hardware to record the address of the current instruction being executed, the addressing mode, and other control information. An example of such a storage report is shown in Figure 2 on page 45, in this case for a program check. Analyze it as follows:

1. Look first in the dump for this header, which introduces the error report for the task:

   ```
   ==KE: KE DOMAIN ERROR TABLE
   ```

2. Next, you will see the kernel error number for the task. Error numbers are assigned consecutively by the kernel, starting from 00000001. You might, for example, see this:

   ```
   =KE: ERROR NUMBER:  00000001
   ```

   The error number tells you the number of program checks and system abends that have occurred for this run of CICS. Not all of them have necessarily resulted in a system dump.

   Some kernel error data follows. If you want to find the format of this data (and, in most cases, you will not need to), see the DFHKERRD section of the *CICS Diagnosis Reference*.

3. The next thing of interest is the kernel's interpretation of what went wrong. This includes the error code, the error type, the name of the program that was running, and the offset within the program.

   The error code gives you the system and user completion codes issued when the abend occurred.

   The error type tells you whether the error was associated with, for example, a program check, a system abend, or an internal request for system recovery.

```
==KE: KE Domain Error Table
=KE: Error Number:  00000001
 KERRD 01F15718 KERNEL ERROR DATA
    0000  F0C3F461 C1D2C5C1 010400C4 0000116E  C4C6C8E3 E2D74040 01273460 01D8B080  *0C4/AKEA...D...>DFHTSP  ...-.Q..*    01F15718
    0020  00891000 011AA380 00000007 00000004  07BD0004 C0000000 07BD0000 812745CE  *.i....t.....................a...*    01F15738
    0040  00060004 00000000 812745CE B0000000  F20005B0 00000000 011AFB80 015748B0  *........a.......2...............*    01F15758
    0060  00000032 81273658 812734CE 012744D1  008E8478 81272B14 011AFEF0 812734D2  *....a...a......J..d.a......0a..K*    01F15778
    0080  00891000 80884D88 011F5028 80884080  00000000 00000000 00000000 00000000  *.i...h(h..&..h ................*    01F15798
    00A0  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    01F157B8
    00C0  00000000 00000000 00000000 00000000  07BD0004 C0000000 07BD0000 812745CE  *.............................a...*    01F157D8
    00E0  00060004 00000000 812745CE B0000000  F20005B0 00000000 011AFB80 015748B0  *........a.......2...............*    01F157F8
    0100  00000032 81273658 812734CE 012744D1  008E8478 81272B14 011AFEF0 812734D2  *....a...a......J..d.a......0a..K*    01F15818
    0120  00891000 80884D88 011F5028 80884080  00000000 00000000 00000000 00000000  *.i...h(h..&..h ................*    01F15838
    0140  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    01F15858
    0160  00000000 00000000 00000000 00000000  B21EB433 0B903880 40404040 40404040  *........................        *    01F15878
    0180  40404040 40404040 40404040 40404040  40404040 40404040 00000000 00000000  *                        ........*    01F15898
 CICS DUMP: SYSTEM=CICSTOR                                                                                    79  13:59:29 99/04/19
    01A0  00000000 00000000                                                         *........                        *    01F158B8
   Error Code:  0C4/AKEA    Error Type:  PROGRAM_CHECK          Timestamp:  B21EB4330B903880
   Date (GMT)   :  19/04/99        Time (GMT)   :  13:57:52.114947
   Date (LOCAL) :  19/04/99        Time (LOCAL) :  13:57:52.114947
   KE_NUM:  0010    KE_TASK:  011AA380    TCA_ADDR:  00891000    DS_TASK:  01D8B080
 Error happened in program DFHTSP   at offset 0000116E
 Registers and PSW.
   PSW:  07BD0000 812745CE   Instruction Length:  6  Interrupt Code:  04  Exception Address:  00000000
   Execution key at Program Check/Abend: B
   REGISTERS 0-15
 REGS 01F15768
    0000  F20005B0 00000000 011AFB80 015748B0  00000032 81273658 812734CE 012744D1  *2...................a...a......J*    01F15768
    0020  008E8478 81272B14 011AFEF0 812734D2  00891000 80884D88 011F5028 80884080  *..d.a......0a..K.i...h(h..&..h .*    01F15788
   Data at PSW: 812745CE    Module: DFHTSP     Offset: 0000116E
 PSWDATA 01273460
    0000  5CC4C6C8 E3E2D740 408168B4 14F0F4F1  F0C91255 02224040 40404040 40400400  **DFHTSP  a...0410I....        ..     01273460
  < data for offset 20 to 1260 follows >
   Data at Registers
   REG 0   F20005B0
 31-bit data cannot be accessed **
 24-bit data follows:
 REGDATA 000005B0
   -0080  00068B00 000C0000 00000000 000074E0  00007570 000089D0 00000000 00359028  *......................i.........*    00000530
   < data for offset -60 to 100 follows >                                                                                  00000550
```

Figure 2. Storage report for a task that has experienced a program check

4. Next, there is a report of where the system has recorded that the error occurred, and the circumstances of the failure. This is the general format of the information:

   ```
   ERROR HAPPENED IN PROGRAM pppppppp AT OFFSET xxxxxxxx
   ```

5. The program name (pppppppp) and offset (xxxxxxxx) are determined by searching through the CICS loader's control blocks for a program that owned the abending instruction at the time of the abend. If this search does not find such a program, the following text appears in the report:

   ```
   PROGRAM pppppppp WAS IN CONTROL, BUT THE PSW WAS ELSEWHERE.
   ```

6. The program name (pppppppp) reported, is the program that owns the current kernel stack entry for the abending task. If this text appears, it may be possible to locate the failing program using the method described in "Using the linkage stack to identify the failing module" on page 47.

7. The failing program name and offset are also displayed in the section of the report immediately after the contents of the registers have been reported. The format of this information is:

   ```
   DATA AT PSW: AAAAAAAA   MODULE: PPPPPPPP   OFFSET: XXXXXXXX
   ```

If the failing program could not be located, the module name and offset are reported as unknown. The possible reasons for the program not being located are:

- The failure occurred in a VSE loaded module
- The failing program had been released by the CICS loader before the dump was taken
- A wild branch in the failing program caused the PSW to point to storage not occupied by a CICS loaded program

Note that the accuracy of the program name and offset cannot be guaranteed when reported in a formatted dump that was produced as the result of a program executing a wild branch.

The error data for the failing task

The error data you get in the task storage report is based on values in the PSW and the registers at the time the error was detected. Figure 2 on page 45 shows the storage report for a task that failed when a program check was detected. It illustrates the error data supplied when an error happens.

The storage addressed by the registers and PSW

Any storage addressed by the registers and PSW is included in the error data for the failing task.

Note: Only the values of the registers and PSW, not the storage they address, are guaranteed to be as they were at the time of the error. The storage that is shown is a snapshot taken at the time the internal system dump request was issued. Data might have changed because, for example, a program check has been caused by an incorrect address in a register, or short lifetime storage is addressed by a register.

Also, in general, where error data is given for a series of errors, the older the error, the less likely it is that the storage is as it was at the time of the failure. The most recent error has the highest error number; it might not be the first error shown in the output.

The registers might point to data in the CICS region. If the values they hold can represent 24-bit addresses, you see the data around those addresses. Similarly, if their values can represent 31-bit addresses, you get the data around those addresses.

It could be that the contents of a register might represent both a 24-bit address and a 31-bit address. In that case, you get both sets of addressed data. (Note that a register might contain a 24-bit address with a higher order bit set, making it appear like a 31-bit address; or it could contain a genuine 31-bit address.)

If, for any reason, the register does not address any data, you see either of these messages:

```
24-bit data cannot be accessed
31-bit data cannot be accessed
```

This means that the addresses cannot be found in the system dump of the CICS region. Such areas are not dumped in a VSE SDUMP of the region.

It is also possible that the addresses were within the CICS region but they were not included in the SDUMP. For example, the address may be within the SVA, and the SVA may not have been included in the list of areas dumped by an SDUMP request. If this were to happen without your knowledge, you might think you had an addressing error when, in fact, the address was a valid one.

The format of the PSW is described in the appropriate *ESA/390 Principles of Operation*. The information in the PSW can help you to find the details needed by the IBM Support Center. You can find the address of the failing instruction, and hence its offset within the module, and also the abend type. You can find the identity of the failing module itself by examining the kernel linkage stack, as described below.

Using the linkage stack to identify the failing module

You can sometimes use the technique described in this section to gather the information that the IBM Support Center needs to resolve the CICS system abend. However, you should normally use the summary information presented in the formatted output for the kernel domain storage areas.

This method is only valid if the abend has occurred in a module or subroutine that has a kernel linkage stack entry. This is the case only where the module or subroutine has been invoked by one of these mechanisms:

- A kernel domain call
- A kernel subroutine call
- A call to an internal procedure identified to the kernel
- A LIFO call

Routines that have been invoked by assembler language BALR instructions do not have kernel linkage stack entries. Having found which task was in error from the kernel's task summary (see "Finding which tasks are associated with the error" on page 42), you need next to find out which module was in error. The module name is one of the things you need to give the IBM Support Center when you report the problem to them.

Find the task number of the task in error from the KE_NUM column, and use this as an index into the linkage stack entries. These are shown in the dump after the task summary.

Figure 3 on page 48 shows what a typical kernel linkage stack looks like.

```
KE_NUM @STACK   LEN  TYPE ADDRESS  LINK REG OFFS ERROR NAME
0031   0520A020 0120 Bot  84C00408 84C006D8 02D0       DFHKETA
0031   0520A140 01F0 Dom  84C0F078 84C0F18E 0116       DFHDSKE
0031   0520A330 0370 Dom  84CAA5A8 84CAACC2 071A       DFHXMTA
0031   0520A6A0 0330 Dom  84F25430 84F25CF6 08C6       DFHPGPG
                     Int     +00CC 84F254B6 0086       INITIAL_LINK
0031   0520A9D0 03C0 Dom  84F6C230 84E5DC40 0000       DFHAPLI1
                     Int     +0EEA 84F6C66E 043E       CICS_INTERFACE
0031   0520AD90 0108 Sub  0230B400 8230B8CA 04CA       DFHEIQSP
0031   0520AE98 0290 Sub  82136D90 82137178 03E8 *YES* DFHLDLD
       0520B128      Int     +08FC 82136F26 0196       LDLD_INQUIRE
       0520B128      Int     +128E 821376CE 093E       CURRENT_GET_NO_WAIT
0031   0520B128 0F70 Dom  84C6F8E0 84C72EA6 35C6       DFHMEME
                     Int     +2CB6 84C6FA4E 016E       SEND
                     Int     +1486 84C72684 2DA4       CONTINUE_SEND
                     Int     +350E 84C70DE4 1504       TAKE_A_DUMP_FOR_CALLER
0031   0520C098 03D0 Dom  84C52458 84C52F52 0AFA       DFHDUDU
                     Int     +08F4 84C5254A 00F2       SYSTEM_DUMP
                     Int     +1412 84C53212 0DBA       TAKE_SYSTEM_DUMP
```

Figure 3. Example of a kernel linkage stack showing a task in error

The TYPE column in the example can contain any of the following entries:

Type	Meaning
Bot	The first entry in the stack.
Dom	Stack entry caused by a domain call.
Sub	Stack entry caused by a subroutine.
Lifo	Stack entry caused by a LIFO module.
Int	Call to an internal procedure identified to the kernel.

A linkage stack for a task represents the sequence in which modules and subroutines have been called during execution of a task. It provides a valuable insight into the sequence of events up until the time of failure, and it also flags any program or subroutine that was executing when the error was detected.

The modules and subroutines are shown in the listing in the order in which they were invoked, so the first module you see is at the bottom of the stack, and the second module is next from bottom. You often see the modules DFHKETA and DFHDSKE, respectively, in these two positions.

The last module or subroutine in the listing is at the top of the stack, and it represents the last call that was made before the dump was taken. Assuming that the system abend caused the dump to be taken, this is likely to be a routine associated with dump domain.

In the example shown, program DFHLDLD is shown to be in error. In this case, DFHLDLD is the module name that you would need to report to the IBM Support Center, together with the other information described in “Using the PSW to find the offset of the failing instruction” on page 49.

Note: If module DFHAPLI is flagged in error, consider first whether an application is to blame for the failure. DFHAPLI is the Application Language Interface Program, and it is on the linkage stack whenever an application is being executed. If an application is the cause of the error, it is your responsibility to correct the problem.

Using the PSW to find the offset of the failing instruction

You can calculate the offset of the failing instruction from the PSW, although in practice you seldom need to because the offset is quoted in the storage report for the task. If you are not sure of the format of the PSW, or how to calculate the offset, see the appropriate *ESA/390 Principles of Operation* manual.

The Support Center also needs to know the instruction at the offset. Locate the address of the failing instruction in the dump, and find out what instruction is there. It is sufficient to give the hex code for the instruction, but make sure you quote as many bytes as you found from the PSW instruction length field.

Identify also the abend type from the program interruption code, so that you can report that, too. It might, for example, be 'protection exception' (interruption code 0004), or 'data exception' (interruption code 0007).

Finding the PTF level of the module in error

The IBM Support Center needs to know the PTF level of any module reported to them as being in error. You can find this in the loader domain program storage map summary, which you can get using the dump formatting keyword LD.

Figure 4 shows some entries from a typical program storage map summary. It shows that program DFHZNEP, for example, is at PTF level ULnnnnn, where nnnnn is a five-digit number.

```
 ==LD: PROGRAM STORAGE MAP
  PGM NAME ENTRY PT  CSECT   LOAD PT. REL. PTF LVL. LAST COMPILED  COPY NO. USERS     LOCN  TYP ATTRIBUTE R/A MODE APE ADDR
                                                                                                 OVERRIDE
  DFHDUIO  8003B000 DFHDUIO  0073B000 0410 ULnnnnn  02/22/99 19.32    1       1       CDSA  ANY REUSABLE  24   31  03AC6330
                    IPRDUIO  0073C358
  DFHCSA   8003CBB0 DFHKELCL 0073C440 0410 ULnnnnn  02/22/99 08.38    1       1       CDSA  RPL RESIDENT   -    -  03AC6430
1                   DFHKELRT 0073C740 0410 ULnnnnn  02/22/99 08.43
                    DFHCSA   0073C9C8 0410 ULnnnnn  I  22/02 10.08
                    DFHCSAOF 0073CF78 0410 ULnnnnn  I  22/02 10.08
                    DFHKERCD 0073D568 0410 ULnnnnn  02/22/99 18.48
                    DFHKERER 0073D738 0410 ULnnnnn  02/22/99 16.28
                    DFHKERRI 0073E0D8 0410 ULnnnnn  02/22/99 16.16
                    DFHKESFM 0073E308 0410 ULnnnnn  02/22/99 16.18
                    DFHKESGM 0073E820 0410 ULnnnnn  02/22/99 16.18
  DFHAIP   0003EE88 DFHEIP   0073EE60 0410 ULnnnnn  I  22/02 17.14    1       2       CDSA  ANY RESIDENT   -    -  03AC6730
                    DFHEIPA  007410A0 0410 ULnnnnn  I  22/02 15.45
                    DFHCPI   00741318 0410 ULnnnnn  I  22/02 18.56
                    DFHAICBP 00741A70 0410 ULnnnnn  I  22/02 16.17
  DFHZNEP  83ADF4C8 DFHYA410 03ADF480 0410 ULnnnnn                    1       0       ECDSA RPL RESIDENT   -    -  03B793B0
```

Figure 4. Part of the loader domain program storage map summary

Note: Entries made in the R/A MODE OVERRIDE columns are the value of the RMODE and AMODE supplied on the DFHLDLD DEFINE_PROGRAM call for that program. If a REQUIRED_RMODE or REQUIRED_AMODE is not specified, a – (dash) symbol appears in the appropriate column. If AMODE_ANY or RMODE_ANY is specified, 'ANY' appears in the appropriate column. Other values are shown as specified.

See the loader domain section of the *CICS Diagnosis Reference* manual for further information on the DEFINE_PROGRAM domain call.

Chapter 6. Dealing with waits

This chapter gives you information about what to do if you are aware that a task is in a wait state during:

- A run
- Initialization
- Termination
- Restart

Refer to Table 4 to find the section of this chapter that you need to read.

Chapter 6, “Dealing with waits” chapter

Table 4. Road map for the

If you want to...	Refer to...
Learn about the ways in which tasks in a CICS system can be made to wait	“How tasks are made to wait” on page 59
Find out about the online and offline techniques to investigate waits	“Techniques for investigating waits” on page 52
Identify the functional area of CICS involved in a wait where you already know the identity of the resource that a task is waiting for	Table 20 on page 69
Find out more about the functional area with which a wait is associated	The section dealing with the functional area, as follows: “Transaction manager waits” on page 78 “Lock manager waits” on page 82 “Spooler waits” on page 77 “Journal control waits” on page 85 “Task control waits” on page 88 “Storage waits” on page 90 “Temporary storage waits” on page 91 “Terminal waits” on page 94 “VTAM terminal control waits” on page 102 “Interregion and intersystem communication waits” on page 104 “Transient data waits” on page 105 “Loader waits” on page 108 “File control waits” on page 108 “Interval control waits” on page 115 “XRF alternate system waits” on page 121 “CICS system task waits” on page 122 “FEPI waits” on page 123
Know what to do in the event of CICS stalling	“What to do if CICS has stalled” on page 124

If you have one or more tasks in a wait state, you should have already carried out preliminary checks to make sure that the problem is best classified as a wait, rather than as a loop or as poor performance. If you have not, you can find guidance about how to do this in “Distinguishing between waits, loops, and poor performance” on page 14.

You are unlikely to have direct evidence that a CICS system task is in a wait state, except from a detailed examination of trace. You are more likely to have noticed that one of your user tasks, or possibly a CICS user task—that is, an instance of a

© Copyright IBM Corp. 1979, 1999

CICS-supplied transaction—is waiting. In such a case, it is possible that a waiting CICS system task could be the cause of the user task having to wait.

For the purpose of this chapter, a task is considered to be in a wait state if it has been suspended after first starting to run. The task is *not* in a wait state if it has been attached to the transaction manager but has not yet started to run, or if it has been resumed after waiting but cannot, for some reason, start running. These are best regarded as performance problems. Tasks that are ready to run but cannot be dispatched might, for example, have too low a priority, or the CICS system might be at the MXT limit, or the CICS system might be under stress (short on storage). If you think you might have such a problem, read Chapter 8, “Dealing with performance problems” on page 139.

Most tasks are suspended at least once during their execution, for example while they wait for file I/O to take place. This is part of the regular flow of control, and it gives other tasks a chance to run in the meantime. It is only when they stay suspended longer than they should that a problem arises.

There are two stages in resolving most wait problems involving user tasks. The first stage involves finding out what resource the suspended task is waiting for, and the second stage involves finding out why that resource is not available. This chapter focuses principally on the first of these objectives. However, in some cases there are suggestions of ways in which the constraints on resource availability can be relieved.

If you know that a CICS system task is in a wait state, it does not necessarily indicate an error in CICS. Some system tasks spend long periods in wait states, while they are waiting for work to do. For more information about waiting system tasks, see “CICS system task waits” on page 122.

Note: Throughout this section, the terms “suspension” and “resumption” and “suspended” and “resumed” are used generically. Except where otherwise indicated, they refer to any of the SUSPEND/RESUME and WAIT/POST processes by which tasks can be made to stop running and then be made ready to run again.

Techniques for investigating waits

You can investigate waits in a CICS system by the following three methods:

- Online inquiry. This is the least powerful technique, and it can only tell you what resource a suspended user task is waiting for. This is enough information to locate the failing area, but you often need to do more investigation before you can solve the problem. The advantage of online inquiry is that you can find out about the waiting task as soon as you detect the problem, and so you capture the data early.
- Tracing. This can give you much more detail than online inquiry, but it involves significant processing overhead. It must also be running with the appropriate options selected when the task first enters a wait state, so this usually means you need to reproduce the problem. However, the information it gives you about system activity in the period leading up to the wait is likely to provide much of the information you need to solve the problem.
- CICS system dump. This gives you a picture of the state of the CICS system at an instant during the wait. You can request the dump as soon as you notice that a task has entered a wait state, so it gives you early data capture for the

problem. The dump is unlikely to tell you anything about system activity in the period leading up to the wait, even if you had internal tracing running with the correct selectivity when the task entered the wait. This is because the trace table has probably wrapped before you have had a chance to respond. However, the formatted dump might contain much of the information you need to solve the problem.

If you are able to reproduce the problem, consider using auxiliary tracing and dumping in combination.

Investigating waits using the online method

Online, you can use either CEMT INQUIRE TASK or EXEC CICS INQUIRE TASK to find out what resource a user task is waiting on. EXEC CICS INQUIRE TASK can be executed under CECI, or issued from a user-written program. Whatever online method you use, you need to supply the task ID of the suspended user task.

If the task is suspended, the information that is returned to you includes the resource type and/or the resource name, identifying the unavailable resource. CEMT INQUIRE TASK displays the resource type of the unavailable resource in the HTYPE field. The HVALUE field displays the resource name of the unavailable resource. EXEC CICS INQUIRE TASK returns values in the SUSPENDTYPE and SUSPENDVALUE fields which correspond to the resource type and resource name of the unavailable resource.

HTYPE and SUSPENDTYPE, and HVALUE and SUSPENDVALUE correspond to the values in the resource type and resource name fields of the dispatcher task summary.

Table 20 on page 69 gives a list of all the resource types and resource names that user tasks might be suspended on, and references showing where to look next for guidance about solving the wait.

You probably need a system dump of the appropriate CICS region to investigate the wait. If you do not yet have one, you can get one using the CEMT PERFORM SNAP or CEMT PERFORM DUMP transactions—but make sure the task is still in a wait state when you take the dump. You subsequently need to format the dump using keywords for the given resource type. Advice on which keywords to use is given, where appropriate, in the individual sections.

Investigating waits using trace

You can find detailed information about the suspension and resumption of tasks during a run of CICS by studying the trace table. Tracing must, of course, be running when the task in question is suspended or resumed, and the tracing options must be selected correctly.

When you look at the trace table, you can find trace entries relating to a particular task from the task numbers that the entries contain. Each is unique to a task so you can be sure that, for any run of CICS, trace entries having the same task number belong to the same task.

For general guidance about setting tracing options and interpreting trace entries, see Chapter 14, “Using traces in problem determination” on page 199.

Setting up trace

The DSSR gate of the dispatcher domain provides the major functions associated with the suspension and resumption of tasks. (See “How tasks are made to wait” on page 59.) The level-1 trace points **DS 0004** and **DS 0005** are produced on entry to, and exit from, the gate.

You need, therefore, to select tracing to capture the DS level-1 trace entries to investigate any wait problem. You need to capture trace entries for other components as well, when you know what functional areas are involved. The functional area invoking the task wait might, for example, be terminal control (TC), or file control (FC). Level-1 tracing is often enough for these components. However, there are cases, such as those relating to VSAM I/O errors where level-2 trace is needed to examine the RPL as it is passed to VSAM.

Next, you need to ensure that tracing is done for the task that has the wait problem. At first select special tracing for just that task, and disable tracing for all other tasks by setting the master system trace flag off. Subsequently, you can select special tracing for other tasks as well if it becomes clear that they are implicated in the wait.

Interpreting trace

For new-style trace entries, which include those for point IDs DS 0004 and DS 0005, the function being traced is shown explicitly in the interpretation string. The functions in Table 7 on page 60 are all provided by the DSSR gate, and you might see any of them traced by these two trace points. The functions that can cause a task to enter a wait state are identified in the table. Look out for these in particular in the trace entries for any waiting task you are investigating.

Each function has its own set of input and output parameters, and these, too, are shown in the interpretation strings of the formatted trace entries. Input parameters are shown in the trace entries made from point ID DS 0004, and output parameters in the trace entries made from point ID DS 0005.

The values of the parameters can provide valuable information about task waits, so pay particular attention to them when you study the trace table.

Investigating waits using the formatted CICS system dump

If you are suitably authorized, you can request a CICS system dump using the CEMT PERFORM DUMP, CEMT PERFORM SNAP, or CECI PERFORM DUMP commands. Make sure that the task in question is waiting when you take the dump, so that you capture information relevant to the wait.

You need to use the dump formatting keyword DS to format the dispatcher task summary. You probably need to look at other areas of the dump as well, so keep the dump on the dump data set.

Interpreting the dump for wait problems

The dispatcher task summary gives you information like that shown in Figure 5 on page 55.

```
===DS: DISPATCHER DOMAIN - SUMMARY


 KEY FOR SUMMARY

     TY = TYPE OF TASK            SY=SYSTEM  NS=NON-SYSTEM
     S  = STATE OF TASK           DIS=DISPATCHABLE  SUS=SUSPENDED
                                  RUN=RUNNING  REE=RESUMED EARLY
     P  = PURGEABLE WAIT/SUSPEND  Y=YES  N=NO
     PS = PURGE STATUS            OK=NO PURGE  PU=PURGED  PP=PURGE PENDING
     TT = TIMEOUT TYPE            IN=INTERVAL  DD=DEADLOCK DELAYED DI=DEADLOCK IMMEDIATE
     ST = SUSPEND TYPE            EXT=WAIT_EXTERNAL  SUSP=SUSPEND  OLDC=WAIT_OLDC  OLDW=WAIT_OLDW
     DTA= DISPATCHER TASK AREA
     AD = ATTACHING DOMAIN
     MO = TASK MODE               QR=QUASI-REENTRANT RO=RESOURCE OWNING  SZ=FEPI OWNING


DS_TOKEN KE_TASK  TY S   P PS TT RESOURCE RESOURCE ST   TIME OF      TIMEOUT      DTA      AD ATTACHER MO SUSPAREA XM_TXN_TOKEN
                                 TYPE     NAME          SUSPEND      DUE          (DSTSK)     TOKEN

00000003 055C9C80 SY SUS N OK -                    SUSP 11:07:46.318 18      -    056FB080 XM 05706300 QR 056FB080 05706300000001
00020003 055D6C80 SY SUS N OK -  TIEXPIRY DS_NUDGE SUSP 11:08:20.291
000C0003 056F0C80 SY SUS N OK -  ICEXPIRY DFHAPTIX SUSP 11:08:20.291
00120005 055C9580 SY SUS N OK -  ICMIDNTE DFHAPTIM SUSP 11:05:01.563
001A0003 055C9200 SY SUS N OK -  TCP_NORM DFHZDSP  OLDW 11:08:20.465
```

Figure 5. Dispatcher task summary

A brief explanation of the summary information is given in the dump. A more detailed explanation is given in the section that follows.

Dispatcher task summary fields

Detailed descriptions of the fields in the dispatcher task summary are given in Table 5. Some of the fields relate to all tasks known to the dispatcher, and some (identified in the table) relate only to suspended tasks. Values are not provided in fields of the latter type for tasks that are not suspended.

Table 5 (Page 1 of 3). Descriptions of fields shown in the dispatcher task summary

Field	Description
AD	The 2-character domain index identifying the domain that attached the task to the dispatcher.
ATTACHER TOKEN	A token provided by the domain that attached the task. This token uniquely identifies the task to the attaching domain.
DS_TOKEN	A token given by the dispatcher to a domain that attaches a task. It identifies the attached task uniquely to the dispatcher.
DTA	An address used internally by the dispatcher.
KE_TASK	A value that uniquely identifies to the kernel a task that has been created.
MO	The dispatching mode for the task. The possible values are: QR—quasi-reentrant. RO—resource owning. SZ—FEPI owning.
P	Whether the suspend call issued by the suspending task specified PURGEABLE(YES) or PURGEABLE(NO). PURGEABLE(NO) inhibits deadlock time-out, CEMT SET TASK PURGE, and EXEC CICS SET TASK PURGE. The possible values are: Y (=YES)—the task is purgeable. N (=NO) —the task is not purgeable.

Table 5 (Page 2 of 3). Descriptions of fields shown in the dispatcher task summary	
Field	**Description**
PS	The purge status of the task. The possible values are: OK—the task has not been purged, and there is no purge pending. PU—the task has been purged, either by the dispatcher or by the operator. PP—there is a purge pending on the task.
RESOURCE NAME *(suspended tasks only)*	The name of the resource that a suspended task is waiting for. A value is given only if RESOURCE_NAME has been included as an input parameter on the suspend call.
RESOURCE TYPE *(suspended tasks only)*	The type of the resource that the task is waiting for. A value is given only if RESOURCE_TYPE has been included as an input parameter on the suspend call.
S	The state of the task within the dispatcher. Possible values are: DIS—the task is dispatchable. It is ready to run, and it will be dispatched when a TCB becomes available. RUN—the task is running. SUS—the task has been suspended by any of the functions SUSPEND, WAIT_EXTERNAL, WAIT_OLDC, or WAIT_OLDW of gate DSSR. For an explanation of these functions, see “How tasks are made to wait” on page 59. REE—the task has been resumed early, possibly because a RESUME request has arrived before the corresponding SUSPEND request. (The SUSPEND/RESUME interface is asynchronous. For more details, see “Function SUSPEND of gate DSSR” on page 62.)
ST *(suspended tasks only)*	The type of function that was invoked to suspend a currently suspended task. Possible values include: EXT—function WAIT_EXTERNAL OLDC—function WAIT_OLDC OLDW—function WAIT_OLDW SUSP—function SUSPEND For a description of the functions, see “How tasks are made to wait” on page 59.
SUSPAREA *(suspended tasks only)*	Either an address used internally by the dispatcher, or an ECB address, or an ECB list address. These are the cases: • Address used internally, if the task was suspended by a SUSPEND call • ECB address or ECB list address, if the task was suspended by a WAIT_EXTERNAL or WAIT_OLDW call • ECB address, if the task was suspended by a WAIT_OLDC call Look at the value given in the ST column to see which one of these descriptions applies.

Table 5 (Page 3 of 3). Descriptions of fields shown in the dispatcher task summary	
Field	**Description**
TIME OF SUSPEND *(suspended tasks only)*	The time when a currently suspended task was suspended. The format is hh:mm:ss.mmm (hours, minutes, seconds, milliseconds), GMT.
TIMEOUT DUE *(suspended tasks only)*	The time that a suspended task is due to timeout, if a timeout interval has been specified. A suspended task only times out if it is not resumed before this time arrives. The format is hh:mm:ss.mmm (hours, minutes, seconds, milliseconds).
TT *(suspended tasks only)*	The time-out type for the task. The possible values, where one is given, are: IN—a time-out interval has been specified for the task. DD—deadlock action is to be delayed when the time-out interval expires. DI—deadlock action is immediate when the time-out interval expires. See "INTERVAL and DEADLOCK_ACTION parameters" on page 67 for more details of time-out interval and deadlock action.
TY	Whether this is a system task or a non-system task. Possible values are: SY—this is a system task. NS—this is a non-system task. A non-system task can be either a user written transaction, or a CICS-supplied transaction.

Parameters and functions setting fields in the dispatcher task summary

Many of the values shown in the dispatcher task summary are provided directly by parameters included on calls to and from the dispatcher. If you are using trace, you can see the values of the parameters in the trace entries, and this can be useful for debugging. For details of how you can use trace to investigate waits, see "Investigating waits using trace" on page 53.

Table 6 shows the parameters that set task summary fields, the functions that use those parameters, and the domain gates that provide the functions. Task summary fields that are *not* set by parameters are also identified (by *none* in "Related parameter" column).

Table 6 (Page 1 of 3). Parameters and functions that set fields shown in the dispatcher task summary				
Field	**Related parameter**	**Function**	**Input or output**	**Gate**
AD	DOMAIN_INDEX	INQUIRE_TASK GET_NEXT	IN OUT	DSBR
DTA	ATTACH_TOKEN	CREATE_TASK	IN	KEDS

Table 6 (Page 2 of 3). Parameters and functions that set fields shown in the dispatcher task summary

Field	Related parameter	Function	Input or output	Gate
DS_TOKEN	TASK_TOKEN	ATTACH CANCEL_TASK PURGE_INHIBIT_QUERY SET_PRIORITY TASK_REPLY GET_NEXT INQUIRE_TASK	OUT IN IN IN IN OUT OUT	DSAT DSBR
KE_TASK	TASK_TOKEN	CREATE_TASK CREATE_TCB PUSH_TASK TASK_REPLY TCB_REPLY	OUT OUT IN IN IN	KEDS
MO	MODE	ATTACH CHANGE_MODE GET_NEXT INQUIRE_TASK	IN IN OUT OUT	DSAT DSBR
P	PURGEABLE	SUSPEND WAIT_EXTERNAL WAIT_OLDC WAIT_OLDW	IN IN IN IN	DSSR
PS	*none*	Not applicable	—	—
RESOURCE NAME	RESOURCE_NAME	ADD_SUSPEND SUSPEND WAIT_EXTERNAL WAIT_OLDC WAIT_OLDW GET_NEXT INQUIRE_TASK	IN IN IN IN IN OUT OUT	DSSR DSBR
RESOURCE TYPE	RESOURCE_TYPE	ADD_SUSPEND SUSPEND WAIT_EXTERNAL WAIT_OLDC WAIT_OLDW GET_NEXT INQUIRE_TASK	IN IN IN IN IN OUT OUT	DSSR DSBR
S *(see note 1)*	STATE	GET_NEXT INQUIRE_TASK	OUT OUT	DSBR
SUSPAREA *(see note 2)*	ECB_ADDRESS *or* ECB_LIST_ADDRESS *(see note 3)*	WAIT_EXTERNAL WAIT_OLDC WAIT_OLDW	IN IN IN	DSSR
TIME OF SUSPEND	*none*	Not applicable	—	—
TASKNO	*none*	Not applicable	—	—
TIMEOUT DUE *(see note 4)*	*none*	Not applicable	—	—

Table 6 (Page 3 of 3). Parameters and functions that set fields shown in the dispatcher task summary				
Field	**Related parameter**	**Function**	**Input or output**	**Gate**
TT	INTERVAL *and* DEADLOCK_ACTION	SUSPEND WAIT_EXTERNAL WAIT_OLDW WAIT_OLDC	IN IN IN IN	DSSR
TY	*none*			
ATTACHER TOKEN	USER_TOKEN	ATTACH PURGE_INHIBIT_QUERY TASK_REPLY GET_NEXT INQUIRE_TASK	OUT OUT	DSAT DSBR
ST	*none*	Not applicable	—	—

Notes:

1. Field S (for STATE) of the dispatcher task summary has a wider range of values than parameter STATE of DSBR functions GET_NEXT and INQUIRE_TASK. Parameter STATE can only have the values READY, RUNNING, or SUSPENDED. For the possible values of field S, see Table 5 on page 55.
2. Parameters ECB_ADDRESS and ECB_LIST_ADDRESS only relate to SUSPAREA when the task has been suspended by the WAIT_EXTERNAL, WAIT_OLDW, or WAIT_OLDC functions of gate DSSR.
3. Parameter ECB_LIST_ADDRESS is only valid for functions WAIT_EXTERNAL and WAIT_OLDW, and not for function WAIT_OLDC.
4. If INTERVAL has been specified, the value of TIMEOUT DUE should be equal to INTERVAL + TIME OF SUSPEND.

How tasks are made to wait

The suspension and resumption of tasks in a CICS system are performed by the dispatcher domain, usually on behalf of some other CICS component. If the exit programming interface (XPI) is being used, it can be at the request of user code.

The major functions associated with the suspension and subsequent resumption of tasks are provided by gate DSSR of dispatcher domain. When a task is to be suspended or resumed, the requesting component calls gate DSSR with the appropriate set of parameters. The required function is included in the parameter list sent on the call, and the corresponding routines are executed by the dispatcher. You can use trace to see the functions that are requested, and the values of parameters that are supplied. See “Investigating waits using trace” on page 53.

Table 7 on page 60 lists the functions provided by the gate, and gives a brief summary of their effect on the status of tasks in the CICS system.

Table 7. Functions provided by dispatcher gate DSSR	
Function	**Effect on status of tasks**
ADD_SUSPEND	None—this does not cause a task to be suspended.
INQUIRE_SUSPEND_TOKEN	None.
DELETE_SUSPEND	None—this does not cause a task to be resumed.
SUSPEND	This can cause a running task to be suspended.
RESUME	This can cause a suspended task to be resumed.
WAIT_EXTERNAL	This can cause a running task to wait.
WAIT_OLDW	This can cause a running task to wait.
WAIT_OLDC	This can cause a running task to wait.

The functions that are significant for problem determination are now described in detail. You should consider their effects, the protocols that must be used, and the parameters they use. If you are using trace, your approach to the wait problem depends on the features of the function that caused the task to wait, so you must find out early in your investigation how the task was suspended.

Some of the functions are available to users through the exit programming interface (XPI). If you have any applications using these XPI functions, make sure that they follow the rules and protocols exactly. For programming information about the XPI, see the *CICS Customization Guide*.

Function ADD_SUSPEND of gate DSSR

The ADD_SUSPEND function of gate DSSR returns a SUSPEND_TOKEN to the calling component. The function is available to users through the exit programming interface (XPI). ADD_SUSPEND does not cause any running task to be suspended.

The ADD_SUSPEND function has only to be called once for any sequence of SUSPEND and RESUME calls for a particular task, because the same token can be used each time.

The input and output parameters for the ADD_SUSPEND function are described in Table 8 and Table 9, respectively.

Table 8. DSSR ADD_SUSPEND input parameters	
Parameter	**Description**
[RESOURCE_NAME] [RESOURCE_TYPE]	Provide default values for SUSPEND calls using this suspend token. The defaults are overridden if values are specified on the SUSPEND call. The parameters are optional, so in some cases you might not see them shown in the formatted DS 0004 and DS 0005 trace entries. A complete list of resource names and types is given in Table 20 on page 69.

Table 9. DSSR ADD_SUSPEND output parameters	
Parameter	**Description**
SUSPEND_TOKEN	The token used to identify a SUSPEND/RESUME pair.
RESPONSE	Possible values are: OK|EXCEPTION|DISASTER|INVALID|KERNERROR For the meanings of RESPONSE values, see Table 19 on page 67.
[REASON]	When RESPONSE is 'DISASTER', REASON has this value: INSUFFICIENT_STORAGE

Function INQUIRE_SUSPEND_TOKEN of gate DSSR

INQUIRE_SUSPEND_TOKEN is used by some CICS system tasks to find out the suspend token provided when the task was created.

The output parameters for the DSSR INQUIRE_SUSPEND_TOKEN function are described in Table 10. (There are no input parameters.)

Table 10. DSSR INQUIRE_SUSPEND_TOKEN output parameters	
Parameter	**Description**
SUSPEND_TOKEN	The token that was provided when the task was created.
RESPONSE	Possible values: OK|DISASTER For the meanings of RESPONSE values, see Table 19 on page 67.

Function DELETE_SUSPEND of gate DSSR

The DELETE_SUSPEND function of gate DSSR discards a SUSPEND_TOKEN. It does not cause any suspended task to be resumed. The function is available to users through the exit programming interface (XPI).

When the SUSPEND_TOKEN has been discarded, it can no longer be used to SUSPEND or RESUME a task. However, note that the dispatcher does not allow a SUSPEND_TOKEN to be discarded while a task is suspended against it, even if the RESUME request is serviced before the SUSPEND request. (This is an asynchronous interface—see "Function SUSPEND of gate DSSR" on page 62.)

The input and output parameters for the DSSR DELETE_SUSPEND function are described in Table 11 and Table 12 on page 62, respectively.

Table 11. DSSR DELETE_SUSPEND input parameter	
Parameter	**Description**
SUSPEND_TOKEN	The suspend token to be discarded.

Table 12. DSSR DELETE_SUSPEND output parameters	
Parameter	**Description**
RESPONSE	Possible values are: OK|EXCEPTION|DISASTER|INVALID|KERNERROR For the meanings of RESPONSE values, see Table 19 on page 67.
[REASON]	When RESPONSE is 'INVALID', REASON can have either of these values: INVALID_SUSPEND_TOKEN—the dispatcher does not recognize the suspend token that has been supplied. SUSPEND_TOKEN_IN_USE —a task is already suspended against the suspend token that has been supplied.

Function SUSPEND of gate DSSR

The SUSPEND function of gate DSSR causes a running task to be suspended. The function is available to users through the exit programming interface (XPI), both explicitly (SUSPEND call) and implicitly (GETMAIN SUSPEND(YES) call).

The task to be suspended is identified by a SUSPEND_TOKEN, which might have been obtained by a previous ADD_SUSPEND call. A task suspended by a DSSR SUSPEND call is resumed by a corresponding DSSR RESUME call, if the task is not purged before the RESUME is issued.

It is important to recognize that this is an asynchronous interface: the RESUME call might have been received before the SUSPEND call. This can be significant if you are attempting to match SUSPEND and RESUME call pairs in the trace table, because you might find the RESUME trace entry was made before the corresponding SUSPEND trace entry. However, there can never be more than one outstanding SUSPEND or RESUME against any suspend token.

The input and output parameters for the DSSR SUSPEND function are described in Table 13 and Table 14 on page 63, respectively.

Table 13 (Page 1 of 2). DSSR SUSPEND input parameters	
Parameter	**Description**
SUSPEND_TOKEN	The suspend token that the task is to be suspended against.
PURGEABLE	The purgeable status of the task. Possible values are: YES|NO A task can be purged by the dispatcher when deadlock is detected, or purged by the operator or an application, if PURGEABLE is YES.
[INTERVAL] [DEADLOCK_ACTION]	INTERVAL and DEADLOCK_ACTION are mutually exclusive alternatives. For a discussion of their significance, see "INTERVAL and DEADLOCK_ACTION parameters" on page 67.

Table 13 (Page 2 of 2). DSSR SUSPEND input parameters

Parameter	Description
[RESOURCE_NAME] [RESOURCE_TYPE]	The name and the type of the resource that the task is to wait for. The parameters are optional, so in some cases you might not see them shown in the formatted DS 0004 and DS 0005 trace entries. A complete list of resource names and types is given in Table 20 on page 69.
[DELAY]	If the ECB is posted within the time specified by the DELAY value, CICS is not to consider the task for dispatch until either the DELAY interval has expired or there is no other work to do.
[TIME_UNIT]	Defines the time units specified on the INTERVAL parameter. Possible values are: SECOND|MILLI-SECOND
[RETRY]	Specifying RETRY(YES) continues the DTIMOUT interval started on the last SUSPEND or WAIT that did not specify RETRY(YES). This allows DTIMOUT to work even if a legitimate loop keeps resuming and then resuspending the task.

Table 14. DSSR SUSPEND output parameters

Parameter	Description
[COMPLETION_CODE]	A completion code to be supplied by the issuer of the RESUME.
RESPONSE	Possible values are: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED For the meanings of RESPONSE values, see Table 19 on page 67.
[REASON]	When RESPONSE is 'PURGED', REASON can have either of these values: TASK_CANCELLED—the task was purged by the operator or an application while it was suspended. TIMED_OUT—the task was automatically resumed because the specified INTERVAL (or the deadlock time-out value specified at task attach) expired. When RESPONSE is 'INVALID', REASON can have any one of these values: INVALID_SUSPEND_TOKEN—the supplied suspend token was not recognized by the dispatcher. ALREADY_SUSPENDED—a SUSPEND call was issued against a suspend token that was already in use. CLEAN_UP_PENDING—the task has automatically been resumed because it has timed out, but resume processing is not yet complete.

Function RESUME of gate DSSR

The RESUME function of gate DSSR causes a suspended task to be resumed. The function is available to users through the exit programming interface (XPI).

The task to be resumed is identified by a SUSPEND_TOKEN, which is the same as the one used to SUSPEND the task in the first instance.

It is important to recognize that this is an asynchronous interface: so the RESUME call might have been received before the SUSPEND call. This can be significant if you are attempting to match SUSPEND and RESUME call pairs in the trace table, because you might find the RESUME trace entry was made before the corresponding SUSPEND trace entry. However, there can never be more than one outstanding SUSPEND or RESUME against any suspend token.

The input and output parameters for the DSSR RESUME function are described in Table 15 and Table 16, respectively.

Table 15. DSSR RESUME input parameters

Parameter	Description
SUSPEND_TOKEN	The suspend token that the task was suspended against.
[COMPLETION_CODE]	A user-supplied completion code.

Table 16. DSSR RESUME output parameters

Parameter	Description
RESPONSE	Possible values are: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED For the meanings of RESPONSE values, see Table 19 on page 67.
[REASON]	When RESPONSE is 'EXCEPTION', REASON can have either of these values: TASK_CANCELLED—the task was purged by the operator or an application while it was suspended. TIMED_OUT—the task was automatically resumed because the specified INTERVAL (or the deadlock time-out value specified at task attach) expired. When RESPONSE is INVALID, REASON can have either of these values: INVALID_SUSPEND_TOKEN—the suspend token was not recognized. ALREADY_RESUMED—the task that was suspended against this suspend token has already been resumed.

Functions WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC of gate DSSR

Functions WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC of gate DSSR have much in common, and they are considered together. Unlike function SUSPEND of gate DSSR, no corresponding function is provided explicitly by gate DSSR to resume a task suspended by any of these functions. Instead, the task

automatically becomes ready to run when an event control block (ECB) is posted. The input and output parameters for the functions are described in Table 17 on page 65 and Table 18 on page 66, respectively.

WAIT_EXTERNAL

The WAIT_EXTERNAL function of gate DSSR causes a task to wait for an ECB, which might be in a list of ECBs, to be posted by the VSE POST macro. The function is available to users through the exit programming interface (XPI). A task made to wait by the WAIT_EXTERNAL function becomes eligible to run again when the single ECB, or any ECB in the list, has been posted.

WAIT_OLDW

The WAIT_OLDW function of gate DSSR causes a task to wait on a single ECB, or list of ECBs. The task becomes eligible to run again when the single ECB, or any ECB in the list, has been posted either by a VSE POST macro or by “hand posting”.

The address of an ECB on a WAIT_OLDW function may have the low order bit ON. This signifies that the task is waiting on a VSE format ECB (posted value is X'00008000') rather than on a CICS or MVS format ECB (posted value is X'40000000'). The bit should be ignored to determine the actual address of an ECB. For example, if a task is waiting on ECB address X'00785641', a task is waiting on a VSE format ECB at address X'00786540'.

WAIT_OLDC

The WAIT_OLDC function of gate DSSR causes a task to wait on a single ECB that must be hand posted. A task made to wait by the WAIT_OLDC function becomes eligible to run again when the ECB has been posted.

Input and output parameters for WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC

Table 17 (Page 1 of 2). DSSR WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC input parameters

Parameter	Description
ECB_ADDRESS ECB_LIST_ADDRESS	These two parameters are mutually exclusive alternatives. ECB_ADDRESS is the address of a single ECB, and ECB_LIST_ADDRESS is the address of a list of addresses. ECB_LIST_ADDRESS is valid only for WAIT_EXTERNAL and WAIT_OLDW.
PURGEABLE	The purgeable status of the task. Possible values are: YES|NO A task can be purged by the dispatcher when deadlock is detected, or purged by the operator or an application, if PURGEABLE is YES.
[INTERVAL] [DEADLOCK_ACTION]	INTERVAL and DEADLOCK_ACTION are mutually exclusive alternatives. For a discussion of their significance, see “INTERVAL and DEADLOCK_ACTION parameters” on page 67.

Table 17 (Page 2 of 2). DSSR WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC input parameters

Parameter	Description
[RESOURCE_NAME] [RESOURCE_TYPE]	The name and the type of the resource that the task is to wait for. The parameters are optional, so in some cases you might not see them shown in the formatted DS 0004 and DS 0005 trace entries. A complete list of resource names and types is given in Table 20 on page 69.
[BATCH] (WAIT_EXTERNAL only)	Whether the requests can be batched. Possible values are: YES|NO
[SPECIAL_TYPE] (WAIT_OLDW only)	Identifies the task as a special type. Its value is always CSTP.
[TIME_UNIT]	The time units used in specifying the INTERVAL value. Can be either seconds or milliseconds.
[DELAY]	If the ECB is posted within the time specified by the DELAY value, CICS is not to consider the task for dispatch until either the DELAY interval has expired or there is no other work to do.
[RETRY]	Specifying RETRY(YES) continues the DTIMOUT interval started on the last SUSPEND or WAIT that did not specify RETRY(YES). This allows DTIMOUT to work even if a legitimate loop keeps resuming and then resuspending the task.
[ECB_EXTENDABLE] (WAIT_EXTERNAL only)	This ECB can be processed by the VSE POST exit DFHDSPX.

Table 18. DSSR WAIT_EXTERNAL, WAIT_OLDW, and WAIT_OLDC output parameters

Parameter	Description
RESPONSE	Possible values are: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED For the meanings of RESPONSE values, see Table 19 on page 67.
[REASON]	When RESPONSE is 'PURGED', REASON can have either of these values: • TASK_CANCELLED—the task was purged by the operator or an application while it was waiting for the ECB to be posted. • TIMED_OUT— either the INTERVAL or the deadlock time-out interval expired. In case of the deadlock time-out, PURGEABLE(YES) must be in effect. When RESPONSE is 'INVALID', REASON can have either of these values: • ALREADY_WAITING—the single ECB specified in ECB_ADDRESS was already being waited on when this WAIT request was received. • INVALID_ECB_ADDR—the ECB address that was supplied is not valid.

INTERVAL and DEADLOCK_ACTION parameters

INTERVAL and DEADLOCK_ACTION are mutually exclusive input parameters on the SUSPEND and WAIT calls to gate DSSR of the dispatcher domain.

INTERVAL is a length of time after which a task is to be resumed automatically by the dispatcher, with a RESPONSE value of 'EXCEPTION' and a REASON value of 'TIMED_OUT'.

DEADLOCK_ACTION describes whether or not the suspended task is to be purged if deadlock is detected, and if so, how it should be purged. The action to be taken depends on the anticipated cause of the deadlock, and it can have any one of these values:

- **DELAYED**
  CICS ensures that it does not deadlock purge more than one task specifying DELAYED in a given interval. This is in the hope that purging one task will free the resources required by the others.
- **IMMEDIATE**
  Deadlock purges of tasks specifying IMMEDIATE go ahead regardless of how many tasks are purged in a given interval. This is used, for example, when purging one task will have no effect on other tasks specifying IMMEDIATE.
- **INHIBIT** This specifies that there is to be no deadlock time-out. INHIBIT overrides any deadlock time-out interval specified at task attach.

The meanings of RESPONSE values returned on DSSR calls

Table 19 explains the meaning of each of the RESPONSE values that can be returned from calls to DSSR functions.

Table 19. Meanings of DSSR function RESPONSE values

RESPONSE	Meaning
OK	When a domain command succeeds, a response of OK is given and the REASON code is not set. The requested function has been completed successfully.
EXCEPTION	Processing of the function could not be completed for the reason specified in the REASON field. If a command fails, the reason for the failure is returned together with an exception response.
DISASTER	The domain could not complete the request because of an unrecoverable system problem.
INVALID	The requested function is not supported by the domain, possibly because the format of the supplied parameters is incorrect.
KERNERROR	The kernel was unable to call the required function gate for the specified REASON.
PURGED	A purge has been requested for the task making the domain call.

The resources that tasks in a CICS system can wait on

Table 20 on page 69 shows all the resources that tasks in a CICS system can wait on. Some resources are identified by both a resource name and a resource type, some by a resource name alone, and some by a resource type alone. The resource names and resource types shown are the ones that you can see in formatted trace entries and, for some resources, by online inquiry.

User tasks can be made to wait only on some of the resources. For each such resource, there is a page reference showing you where to look for guidance about dealing with the wait.

The remaining resources are used only by CICS system tasks. If you have evidence that a system task is waiting on such a resource, and it is adversely affecting the operation of your system, contact your IBM Support Center, after first reading “CICS system task waits” on page 122.

Table 20 (Page 1 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
(none)	DMWTQUEU	DFHDMWQ	SUSPEND	System only	"CICS system task waits" on page 122
(none)	LMQUEUE	DFHLMLM	SUSPEND	User	"Lock manager waits" on page 82
ADAPTER	FEPI_RQE	DFHSZATR	WAIT_ EXTERNAL	User	*See note 6 on page 77*
ALLOCATE	TCTTETI value	DFHALP	SUSPEND	User	"Interregion and intersystem communication waits" on page 104
Any_MBCB	transient data queue name	DFHTDSUB	SUSPEND	User	"Transient data waits" on page 105
Any_MRCB	transient data queue name	DFHTDSUB	SUSPEND	User	"Transient data waits" on page 105
AP_INIT	CSADLECB	DFHSII1	WAIT_OLDC	System only	"CICS system task waits" on page 122
AP_INIT	ECBTCP	DFHAPSIP	WAIT_OLDC	System only	"CICS system task waits" on page 122
AP_INIT	SIPDMTEC	DFHAPSIP	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
AP_INIT	TCTVCECB	DFHSII1	WAIT_OLDC	System only	"CICS system task waits" on page 122
AP_QUIES	CSASSI2	DFHSTP	WAIT_OLDC	System only	"CICS system task waits" on page 122
AP_QUIES	SHUTECB	DFHSTP	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
AP_TERM	STP_DONE	DFHAPDM	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
CCSTWAIT	VSMSTRNG	DFHCCCC	WAIT_OLDC	System only	"CICS system task waits" on page 122
CCVSAMWT	ASYNRESP	DFHCCCC	WAIT_OLDW	System only	"CICS system task waits" on page 122
CCVSAMWT	EXCLOGER	DFHCCCC	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
CDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
CEPWTERM	DFHPSPCK	DFHPSPCK	WAIT_OLD	System only	"CICS system task waits" on page 122
CONSP	DFHPSPIO	DFHPSPIO	WAIT_OLDW	User	"Spooler waits" on page 77
DFHAIIN	AITM	DFHAIIN1	SUSPEND	System only	"CICS system task waits" on page 122
DFHCPIN	CPI	DFHCPIN1	SUSPEND	System only	"CICS system task waits" on page 122

Table 20 (Page 2 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
DFHPRIN	PRM	DFHPRIN1	SUSPEND	System only	"CICS system task waits" on page 122
DFHSIPLT	EARLYPLT	DFHSII1	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
DFHSIPLT	LATE_PLT	DFHSIJ1 or DFHS1PLT	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
ECDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
EDF	DBUGUSER	DFHEDFX	SUSPEND	User	"Task control waits" on page 88
EKCWAIT	Value of NAME argument on EXEC CICS WAIT EVENT command	DFHEKC	WAIT_OLDW	User	"Task control waits" on page 88
ERDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
ESDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
EUDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
FCBFWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FCDWWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FCFRWAIT	fileID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FCFSWAIT	file ID	DFHFCFS	WAIT_OLDC	User	"File control waits" on page 108
FCINWAIT	STATIC	DFHFCIN1	WAIT_OLDC	System only	"CICS system task waits" on page 122
FCIOWAIT	file ID	DFHFCBD or DFHFCVR	WAIT_OLDW	User	"File control waits" on page 108
FCPSWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FCRBWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FCSRSUSP	file ID	DFHFCVR	SUSPEND	User	"File control waits" on page 108
FCTISUSP	file ID	DFHFCVR	SUSPEND	User	"File control waits" on page 108
FCXCWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108

Table 20 (Page 3 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
FCCIWAIT	file ID	DFHFCVR	WAIT_OLDC	User	"File control waits" on page 108
FEPRM	SZRDP	DFHSZRDP	WAIT_ EXTERNAL	CSZI	*See note 6 on page 77*
FOREVER	DFHXMTA	DFHXMTA	WAIT_ EXTERNAL	User	"A user task is waiting on resource type FOREVER" on page 82
ICEXPIRY	DFHAPTIX	DFHAPTIX	SUSPEND	System only	"CICS system task waits" on page 122
ICGTWAIT	terminal ID	DFHICP	SUSPEND	User	"Interval control waits" on page 115
ICMIDNTE	DFHAPTIM	DFHAPTIM	SUSPEND	System only	"CICS system task waits" on page 122
ICWAIT	terminal ID *(See note 1 on page 77)*	DFHICP	SUSPEND	User	"Interval control waits" on page 115
IO_WAIT	DUIO_WAIT	DFHDUIO	WAIT_OLDW	System only	"CICS system task waits" on page 122
CSNC	MROQUEUE	DFHCRNP	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
IRLINK	SYSIDNT concatenated with session name	DFHZIS2	WAIT_ EXTERNAL	User	"Terminal waits" on page 94
JACDIORB	DFHJAP	DFHJAP	WAIT_OLDW	System only	"CICS system task waits" on page 122
JASUBTAS	JASTMECB	DFHJCSDJ	WAIT_OLDC	User	"Journal control waits" on page 85
JCAVLECB	LECBECB	DFHJCP	WAIT_OLDC	User	"Journal control waits" on page 85
JCBUFFER	AVAIL_nn *(See note 2 on page 77)*	DFHJCP	WAIT_OLDC	User	"Journal control waits" on page 85
JCBUFFER	JCTBAECB	DFHJCKOJ	WAIT_OLDC	System only	"CICS system task waits" on page 122
JCBUFFER	JCTBAECB	DFHJCSDJ	WAIT_OLDC	User	"Journal control waits" on page 85
JCCLDONE	SUBTASK	DFHJCC	WAIT_OLDW	User	"Journal control waits" on page 85
JCDETACH	SUBTASK	DFHJCSDJ	WAIT_OLDC	User	"Journal control waits" on page 85
JCFLBUFF	FLUSH_nn *(See note 2 on page 77)*	DFHJCP	WAIT_OLDC	User	"Journal control waits" on page 85

Table 20 (Page 4 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
JCINITN	JOURNALS	DFHJCP	WAIT_OLDC	System only	"CICS system task waits" on page 122
JCIOBLOK	Jnnbbbbb *(See notes 2 on page 77 and 3 on page 77)*	DFHJCP	WAIT_OLDC	User	"Journal control waits" on page 85
JCIOCOMP	JCTICA	DFHJCI or DFHJCP	WAIT_OLDW	System only	"CICS system task waits" on page 122
JCIOCOMP	JCTIOECB	DFHJCC	WAIT_OLDC	User	"Journal control waits" on page 85
JCIOCOMP	RDBLOKnn *(See note 2 on page 77)*	DFHJCO	WAIT_OLDW	System only	"CICS system task waits" on page 122
JCIOCOMP	RDRECDnn *(See note 2 on page 77)*	DFHJCO	WAIT_OLDW	System only	"CICS system task waits" on page 122
JCJASUS	JABSUTOK	DFHJASP	SUSPEND	System only	"CICS system task waits" on page 122
JCJOURDS	DLBL filename	DFHJCP	SUSPEND	System only	"CICS system task waits" on page 122
JCLASTBK	Jnnbbbbb *(See notes 2 and 3 on page 77)*	DFHJCEOV	WAIT_OLDC	User	"Journal control waits" on page 85
JCOPDONE	SUBTASK	DFHJCO	WAIT_OLDW	User	"Journal control waits" on page 85
JCREADY	JCTXAECB	DFHJCO	WAIT_OLDW	User	"Journal control waits" on page 85
JCREADY	JCTXBECB	DFHJCO	WAIT_OLDW	User	"Journal control waits" on page 85
JCRQDONE	SUBTASK	DFHJCC DFHJCKOJ DFHJCO	WAIT_OLDW WAIT_OLDW WAIT_OLDW	User System only User	"Journal control waits" on page 85
JCSWITCH	DS_SW_nn *(See note 2 on page 77)*	DFHJCP	WAIT_OLDC	System only	"CICS system task waits" on page 122
JCSWITCH	DS_SDJnn *(See note 2 on page 77)*	DFHJCSDJ	WAIT_OLDC	User	"Journal control waits" on page 85
JCTAPE2	PREOPNnn *(See note 2 on page 77)*	DFHJCEOV	WAIT_OLDC	User	"Journal control waits" on page 85
JCTAPE2	STABLEnn *(See note 2 on page 77)*	DFHJCSDJ	WAIT_OLDC	User	"Journal control waits" on page 85

Table 20 (Page 5 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
JCTERMN	SUBTASK	DFHJCBSP	WAIT_OLDW	System only	"CICS system task waits" on page 122
JCTIMER	DFHJ1Snn *(See note 2 on page 77)*	DFHJCP	WAIT_OLDC	System only	"CICS system task waits" on page 122
KCCOMPAT	CICS	DFHXCPA	WAIT_OLDC	User	"Task control waits" on page 88
KCCOMPAT	LIST	DFHXCPA	WAIT_OLDW	User	"Task control waits" on page 88
KCCOMPAT	SINGLE	DFHXCPA	WAIT_OLDW	User	"Task control waits" on page 88
KCCOMPAT	TERMINAL	DFHXCPA	SUSPEND	User	"Task control waits" on page 88, and "Terminal waits" on page 94
KC_ENQ	SUSPEND	DFHXCPC	SUSPEND	User	"Task control waits" on page 88, "Transient data waits" on page 105, and "File control waits" on page 108
KERNEL	DETACH	DFHKETA	SUSPEND	User	"Loader waits" on page 108
MBCB_xxx *(See note 4 on page 77)*	transient data queue name	DFHTDSUB	SUSPEND	User	"Transient data waits" on page 105
MXT	XM_HELD	DFHXMAT	*(See note 5 on page 77)*	User	"Maximum task condition waits" on page 78
OPEN_ANY	DFHPSPIO	DFHPSPIO	WAIT_OLDW	User	"Spooler waits" on page 77
OPEN_NFY	DFHPSPIO	DFHPSPIO	WAIT_OLDW	User	"Spooler waits" on page 77
PROGRAM	program ID	DFHLDLD	SUSPEND	User	"Loader waits" on page 108
PROGRAM	program_name	DFHLDDMI	SUSPEND	User	"Loader waits" on page 108

Table 20 (Page 6 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
PROGRAM	program name	DFHPGAI DFHPGCM DFHPGDD DFHPGEX DFHPGIS DFHPGLD DFHPGLE DFHPGLK DFHPGLU DFHPGPG DFHPLRP DFHPGXE	SUSPEND	User	"Loader waits" on page 108
PS_INIT	CSAOPFRC	DFHPSP	WAIT_OLDC	System only	"CICS system task waits" on page 122
RDSA	(none)	DFHSMSQ	SUSPEND	User	
"Storage waits" on page 90					
RECEIVE	DFHPSPIO	DFHPSPIO	WAIT_OLDW	User	"Spooler waits" on page 77
SDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
SMSYSTEM	(none)	DFHSMSY	SUSPEND	System only	"CICS system task waits" on page 122
STARTUP	TSMCPECB	DFHRCRP	WAIT_OLDC	System only	"CICS system task waits" on page 122
SUBTASK	SISUBECB	DFHRCRP	WAIT_ EXTERNAL	System only	"CICS system task waits" on page 122
SUCNSOLE	WTO	DFHSUWT	WAIT_OLDW	System only	"CICS system task waits" on page 122
TCLASS	tranclass name	DFHXMCL	*(See note 7 on page 77)*	User	"Transaction manager waits" on page 78
TCP_NORM	DFHZDSP	DFHZDSP	WAIT_OLDW	System only	"CICS system task waits" on page 122
TCP_SHUT	DFHZDSP	DFHZDSP	WAIT_OLDW	System only	"CICS system task waits" on page 122
TCTVCECB	ZC_ZGRP	DFHZGRP	WAIT_OLDW	System only	"CICS system task waits" on page 122
TDEPLOCK	transient data queue name	DFHTDEXP	SUSPEND	User	"Transient data waits" on page 105
TD_INIT	DCT	DFHTDA	SUSPEND	User	"Transient data waits" on page 105

Table 20 (Page 7 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
TDIPLOCK	transient data queue name	DFHTDB	SUSPEND	User	"Transient data waits" on page 105
TDQ_IORB	transient data queue name	DFHTDSUB	WAIT_OLDW	User	"Transient data waits" on page 105
TIEXPIRY	DS_NUDGE	DFHTISR	SUSPEND	System only	"CICS system task waits" on page 122
TRANDEF	transaction ID	DFHXMDD DFHXMQD DFHXMXD	SUSPEND	System only	"Transaction manager waits" on page 78
TSAUX	(none)	DFHTSP	SUSPEND	User	"Temporary storage waits" on page 91
TSBUFFER	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
TSEXTEND	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
TSIO	(none)	DFHTSP	WAIT_OLDW	User	"Temporary storage waits" on page 91
TSMCPECB	STARTUP	DFHRCRP	WAIT_OLDW	System	"CICS system task waits" on page 122
TSOPEN4B	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
TSQUEUE	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
TSSTRING	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
TSUT	(none)	DFHTSP	WAIT_OLDW WAIT_OLDC	User	"Temporary storage waits" on page 91
TSWBUFFR	(none)	DFHTSP	WAIT_OLDC	User	"Temporary storage waits" on page 91
UDSA	(none)	DFHSMSQ	SUSPEND	User	"Storage waits" on page 90
USERWAIT	ECB list	DFHEIQSK	WAIT_OLDW	User	"Task control waits" on page 88
XPCCWAIT	DFHPSPIO	DFHPSPIO	WAIT_OLDW	User	"Spooler waits" on page 77
WAITLIST	DFHEVBBF	DFHEVBBF	WAIT_OLDW	User	"Task control waits" on page 88
WTOR_RPY	DFHXRCP	DFHXRCP	WAIT_OLDW	System	"CICS system task waits" on page 122
WTOR_RPY	DFHEVBBF	DFHEVBBF	WAIT_OLDW	User	"Task control waits" on page 88
XOVR_RESP	WCSTECB	DFHWSRTR	WAIT_OLDW	System	"CICS system task waits" on page 122

Table 20 (Page 8 of 9). Resources that a suspended task might be waiting on

Resource type	Resource name	Suspending module	DSSR call	Task	Where to look next
XRF_SYNC	XRSSTECB	DFHXRA	WAIT_OLDC	System	"XRF alternate system waits" on page 121
XRF_WAIT	DFHXRA	DFHXRA	WAIT_OLDC	System	"XRF alternate system waits" on page 121
XRGETMSG	message queue name	DFHWMQG	WAIT_OLDW	System only	"CICS system task waits" on page 122
XRPUTMSG	message queue name	DFHWMQP	WAIT_OLDW	User	"XRF alternate system waits" on page 121
ZC	DFHZRAR1	DFHZRAR	SUSPEND	System only	"VTAM terminal control waits" on page 102
ZC	DFHZRAQ1	DFHZRAQ	SUSPEND	System only	"VTAM terminal control waits" on page 102
ZC	DFHZCRQ1	DFHZCRQ	SUSPEND	System only	"How tasks are made to wait" on page 59
ZC	DFHZNAC1	DFHZNAC	SUSPEND	System only	"How tasks are made to wait" on page 59
ZC	DFHZEMU1	DFHZEMU1	SUSPEND	System only	"VTAM terminal control waits" on page 102
ZC	DFHZIS11	DFHZIS1	SUSPEND	System only	"VTAM terminal control waits" on page 102
ZC_ZCGRP	ZSLSECB	DFHZCGRP	WAIT_OLDW	System only	"VTAM terminal control waits" on page 102
ZC_ZGRP	PSINQECB	DFHZGRP	WAIT_OLDW	System only	"VTAM terminal control waits" on page 102
ZC_ZGRP	PSOP1ECB	DFHZGRP	WAIT_OLDW	System only	"VTAM terminal control waits" on page 102
ZC_ZGRP	PSOP2ECB	DFHZGRP	WAIT_OLDW	System only	"VTAM terminal control waits" on page 102
ZC_ZGRP	TCTCVECB	DFHZGRP	WAIT_OLDC	System only	"VTAM terminal control waits" on page 102
ZC_ZGUB	PSUNBECB	DFHZGUB	WAIT_OLDC	System only	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZERH1	DFHZERH	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZERH2	DFHZERH	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZERH3	DFHZERH	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCZNAC	DFHZERH4	DFHZERH	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZARER	DFHZARER	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZARL1	DFHZARL	SUSPEND	User	"VTAM terminal control waits" on page 102

Table 20 (Page 9 of 9). Resources that a suspended task might be waiting on					
Resource type	**Resource name**	**Suspending module**	**DSSR call**	**Task**	**Where to look next**
ZCZGET	DFHZARL2	DFHZARL	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCZNAC	DFHZARL3	DFHZARL	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZARL4	DFHZARL	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZARQ1	DFHZARQ	SUSPEND	User	"VTAM terminal control waits" on page 102
ZCIOWAIT	DFHZARR1	DFHZARR1	SUSPEND	User	"VTAM terminal control waits" on page 102
ZXQOWAIT	LIST	DFHZXQO	WAIT_OLDW	System	"VTAM terminal control waits" on page 102
ZXSTWAIT	LIST	DFHZXST	WAIT_OLDW	System	"VTAM terminal control waits" on page 102

Notes:

1. If there is a terminal associated with the task.
2. "nn" is the two-character journal ID.
3. "bbbbb" is the block number.
4. "xxx" is literal.
5. The task has not yet started, because the system is at MXT (maximum tasks in CICS system).
6. These waits are used by the CICS Front End Programming Interface and are not discussed in this manual.
7. The task has not yet started because it is being held for transaction class purposes.

Spooler waits

Read this section if the resource name your task is waiting on is DFHPSPIO.

Resource type CONSP

When CICS issues a command to VSE/POWER it establishes a communication path via the VSE/ESA XPCC macro. This wait ensures that VSE/POWER has responded positively to the request for communication before proceeding.

Resource type OPEN_ANY

CICS receives order request commands from VSE/POWER through the VSE/ESA XPCC macro. This wait is issued to ensure that the connection between CICS and VSE/POWER has completed before CICS requests VSE/POWER to send device orders.

Resource type OPEN_NFY

CICS receives messages from VSE/POWER through the VSE/ESA XPCC macro. This wait is associated with a 30-second timer. If VSE/POWER does not connect within 30 seconds, CICS issues message DFHRC5482 and tries to establish a connection with VSE/POWER again.

Resource type RECEIVE

CICS has asked VSE/POWER to pass data to it. This wait ensures that VSE/POWER has completed the data transfer before processing the data.

Resource type XPCCWAIT

After a printer has finished printing a report, CICS waits for another suitable report for the printer, or for the printer's timeout value to expire.

Transaction manager waits

Formatting a system dump using the keyword XM=1 provides a number of transaction manager summaries that are useful for identifying why tasks have failed to run.

A task may fail to run if the system is at MXT, or if the task is defined in a transaction class that is at its MAXACTIVE (maximum tasks in transaction class) limit.

Maximum task condition waits

Tasks can fail to run if either of the following limits is reached:

- MXT
- MAXACTIVE

If a task is waiting for entry into the MXT set of transactions, the resource type is MXT, and the resource name is XM_HELD. If a task is waiting for entry into the MAXACTIVE set of transactions for a TRANCLASS, the resource type is TCLASS, and the resource name is the name of the TRANCLASS that the task is waiting for.

If a task is shown to be waiting on resource type MXT, it is being held by the transaction manager because the CICS system is at the MXT limit. The task has not yet been attached to the dispatcher.

The limit that has been reached, MXT, is given explicitly as the resource name for the wait. If this type of wait occurs too often, consider changing the MXT limit for your CICS system.

Transaction summary

The transaction summary (Figure 6 on page 80) lists all transactions (user and system) that currently exist. The transactions are listed in order of task number and the summary contains two lines per transaction.

The meanings of the column headings are as follows:

Tran id The primary transaction ID associated with the transaction

Tran num The unique transaction number assigned to the transaction

Txn Addr The address of the transaction control block

Txd Addr The address of the transaction definition instance associated with the transaction

Start Code The reason the transaction was attached, as follows:

- **C** A CICS internal attach
- **T** A terminal input attach
- **TT** A permanent transaction terminal attach
- **QD** A transient data trigger level attach
- **S** A START command without any data
- **SD** A START command with data
- **SZ** A front end programming interface (FEPI) attach
- **DF** Start code not yet known–to be set later.

Sys Tran Indicator (Yes or No) of whether the transaction is attached as a system transaction. System transactions do not contribute towards MXT.

Status An indicator of how far through attach the transaction has progressed and whether the transaction is abending or not. The first line may take the following values:

- **PRE** The transaction is in the early stages of attach.
- **TCLASS** The transaction is waiting to acquire membership of a transaction class.
- **MXT** The transaction is waiting on MXT.
- **ACT** The transaction is active, that is, it has been DS attached.

Depending on the value in the first line, the second line of the status field may further qualify the transaction state. For each first line value, the meaning of the second line is as follows:

- **PRE** No data is displayed in the second line
- **TCLASS** The second line contains the name of the transaction class that the transaction is waiting to join.
- **MXT or ACT** If applicable, the second line indicates if the transaction is flagged for deferred abend or a deferred message, or if the transaction is already abending, as follows:
  - **DF(xxxx)** indicates that the transaction is scheduled for deferred abend, where xxxx is the abend code.
  - **DM(yy)** indicates that the transaction is scheduled for a deferred message, and yy indicates the message type
  - **AB(xxxx)** indicates that the transaction is already abending with abend code xxxx.

DS token The token identifying the DS task (if any) assigned to the transaction.

Facility type Type of the principal facility owned by the transaction.

Facility token Transaction token for the principal facility owner.

AP token The AP domain transaction token.

The first word of this token contains the address of the TCA (if any) associated with the transaction.

PG token The program manager transaction token.

XS token The security domain transaction token.

US token The user domain transaction token.

RM token The recovery manager transaction token.

SM token The storage manager domain transaction token.

MN token The monitoring domain transaction token.

Figure 6 shows a transaction summary.

```
==XM: TRANSACTION SUMMARY

Tran Tran  TxnAddr Start Sys  Status   DS       Facility Facility AP       PG       XS       US       RM       SM       MN
id   num   TxdAddr code  Tran          token    type     token    token    token    token    token    token    token    token
-------------------------------------------------------------------------------------------------------------------------------
CSTP 00003 10106200 C    Yes  ACT      00120003 None     n/a      10164600 00000000 00000000 00000000 1016C000 10089020 00000000
           101793C0                                               01000000 1017E000 00000000 00000000 10164600 00000000 00000000

CSNE 00031 10106100 C    Yes  ACT      00000003 None     n/a      10164C00 00000000 00000000 00000000 1016C058 11542054 00000000
           10A34B40                                               01000000 1017E048 00000000 00000000 10164C00 00000000 00000000

IC06 10056 10E2B200 T    No   ACT      089601C7 Terminal 10E167A0 1124F600 00000000 00000000 10114023 1016C9A0 11543610 00000000
           10AC9300                                               00000000 00000000 1017E7E0 00000000 10E0F6A0 1124F600 00000000 00000000

IC12 10058 10E34C00 SD   No   ACT      050601AD None     n/a      001DE600 00000000 00000000 10114023 1016C9F8 11545114 00000000
           10AC93C0                                               00000000 1017E828 00000000 10E31400 001DE600 00000000 00000000

TA03 93738 10E0E000 T    No   ACT      088211E3 Terminal 10ED9000 0024B000 00000000 00000000 10114023 1016C738 115437B0 00000000
           10AD3D40                                               00000000 00000000 1017E090 00000000 10117D60 0024B000 00000000 00000000

TA03 93920 10AFF200 T    No   TCLASS   00000000 Terminal 11214BD0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3D40          DFHTCL03                    00000000 00000000 00000000 00000000 10117680 00000000 00000000 00000000

TA03 93960 10E2D200 T    No   TCLASS   00000000 Terminal 10E573F0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3D40          DFHTCL03                    00000000 00000000 00000000 00000000 10E0F6C0 00000000 00000000 00000000

TA03 93967 10AFEA00 T    No   TCLASS   00000000 Terminal 10ECCBD0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3D40          DFHTCL03                    00000000 00000000 00000000 00000000 10117540 00000000 00000000 00000000

TA03 94001 10E34800 T    No   ACT      00000000 Terminal 10E2C3F0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3D40          DF(AKCC)                    00000000 00000000 00000000 00000000 10E31120 00000000 00000000 00000000

TA02 95140 10E2D300 T    No   ACT      0386150D Terminal 10E2C5E8 00057000 00000000 00000000 10114023 1016C790 11544754 00000000
           10AD3C80                                               00000000 00000000 1017E510 00000000 10E0F320 00057000 00000000 00000000

TA02 95175 10E12C00 T    No   TCLASS   00000000 Terminal 10E937E0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3C80          DFHTCL02                    00000000 00000000 00000000 00000000 10E0F100 00000000 00000000 00000000

TA02 95187 10E0B000 T    No   TCLASS   00000000 Terminal 10EA95E8 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3C80          DFHTCL02                    00000000 00000000 00000000 00000000 10117800 00000000 00000000 00000000

TA02 95205 10E2D600 T    No   MXT      00000000 Terminal 10E837E0 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3C80          DF(AKCC)                    00000000 00000000 00000000 00000000 10E0F780 00000000 00000000 00000000

TA04 96637 10E33000 T    No   ACT      060408E7 Terminal 10E05BD0 00057600 00000000 00000000 10114023 1016C7E8 115457C8 00000000
           10AD3E00                                               00000000 00000000 1017E558 00000000 10E31040 00057600 00000000 00000000

TA04 96649 10E34000 T    No   TCLASS   00000000 Terminal 10AE89D8 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD3E00          DFHTCL04                    00000000 00000000 00000000 00000000 10E312C0 00000000 00000000 00000000

F121 99305 10E2D800 T    No   ACT      020C1439 Terminal 10EA93F0 00060000 00000000 00000000 10114023 1016C898 115423FC 00000000
           10AD3BC0          AB(AFCY)                    00000000 00000000 1017E708 00000000 10E0F920 00060000 00000000 00000000

TS12 99344 10AFED00 T    No   MXT      00000000 Terminal 10E499D8 00000000 00000000 00000000 10114023 00000000 00000000 00000000
           10AD6B40                                               00000000 00000000 00000000 00000000 101178C0 00000000 00000000 00000000
```

Figure 6. Transaction summary

Notes for Figure 6

- Transactions 00003 and 00031 are system transactions.
- Transactions 93920, 93960, and 93967 are waiting because transaction class DFHTCL03 is at its MAXACTIVE limit.
- Transaction 94001 is scheduled to abend AKCC because it was attached when transaction class DFHTCL03 was at its PURGETHRESH limit.
- Transactions 95175 and 95187 are waiting because transaction class DFHTCL02 is at its MAXACTIVE limit.
- Transaction 95205 was scheduled to abend AKCC because it was attached when transaction class DFHTCL02 was at its PURGETHRESH limit. It was subsequently made to queue because CICS is at its MXT limit.
- Transaction 99305 is abnormally terminating with abend code AFCY.
- Transaction 99344 is queuing because the system is at its MXT limit.
- Transactions waiting in the transaction manager have no DS token, which is indicated by zeros in the summary.

MXT summary

The MXT summary indicates whether CICS is currently at MXT and shows the current number of queued and active transactions. To check the status of an individual transaction, consult the main transaction summary (Figure 6 on page 80).

```
==XM: MXT SUMMARY

   Maximum user tasks (MXT):                7

   System currently at MXT:                 Yes

   Current active user tasks:               7

   Current queued user tasks:               2

 * Peak active user tasks:                  7

 * Peak queued user tasks:                  2

 * Times at MXT limit:                      1

* NOTE: these values were reset at 18:00:00 (the last statistics interval collection)
```

Transaction class summary

The transaction class summary lists each transaction class that is currently installed. For each class, the current number of active and queued transactions is shown. A transaction class is at its MAXACTIVE limit if its ‘current active’ total is greater than or equal to its ‘max active’ setting. If a transaction class is at its MAXACTIVE limit, a number of transactions may be queueing in that transaction class. The transaction id and number of each queued transaction is listed with its transaction class (for example, transaction classes DFHTCL01, DFHTCL02, DFHTCL03, and DFHTCL04 in Figure 7 on page 82).

```
==XM: TCLASS SUMMARY

 Tclass       Max     Purge    Current   Current     Total    Queuing  Queuing      Queuing
  Name      Active  Threshld    Active    Queued  Attaches    TranNum  Transid     Start Time
--------  --------  --------  --------  --------  --------    -------   ----     ------------
DFHTCL01         1         0         0         0         0
DFHTCL02         1         3         1         2         7    95175     TA02     18:00:19.677
                                                             95187     TA02     18:00:24.624
DFHTCL03         1         4         1         3        29    93920     TA03     17:55:40.584
                                                             93960     TA03     17:55:42.230
                                                             93967     TA03     17:55:52.253
DFHTCL04         1         0         1         1        23    96649     TA04     18:06:04.348
DFHTCL05         1         0         0         0         0
DFHTCL06         1         0         0         0         0
DFHTCL07         1         0         0         0         0
DFHTCL08         1         0         0         0         0
DFHTCL09         1         0         0         0         0
DFHTCL10         1         0         0         0         0

*** Note that the 'Total Attaches' figures were reset at 18:00:00 (the last statistics interval collection)
```

Figure 7. Transaction class summary

A user task is waiting on resource type FOREVER

If you have found that a user task is waiting on a resource type of FOREVER, and resource name DFHXMTA, transaction manager has detected a severe error during task initialization or task termination. Transaction manager has suspended the task.

The suspended task is never resumed, and holds its MXT slot until CICS is terminates. **You must cancel CICS to remove this task as you will be unable to quiesce the system**. You cannot purge or forcepurge the task.

This wait is always preceded by one of the following messages: DFHXM0303, DFHXM0304, DFHXM0305, DFHXM0306, DFHXM0307, DFGXM0308, DFHXM0309, DFHXM0310. Transaction manager also takes a dump and message DFHME0116 is produced and contains the symptom string.

Lock manager waits

Read this section if a resource name of LMQUEUE has been shown for a task. It means that the suspended task cannot acquire the lock on a resource it has requested, probably because another task has not released it.

A user task cannot explicitly acquire a lock on a resource, but many of the CICS modules that run on behalf of user tasks do lock resources. If this is a genuine wait, and the system is not just running slowly, this could indicate a CICS system error.

Collecting the evidence

You need to take a system dump, and format it using keywords LM and DS. This formats the storage areas belonging to lock manager domain and dispatcher domain. This section describes the data that you should find there if the resource locks are being managed correctly.

Turn to the lock manager summary information (Figure 8 shows an example of this).

```
LOCK      LOCK       OWNER       MODE    COUNT   # LOCK     # LOCK    -> QUEUE
NAME      TOKEN                                  REQUESTS   SUSPENDS
----      -----      -----       ----    -----   --------   --------  --------
SMLOCK    03B051D8                                      0          0
DSITLOCK  03B05208                                      4          0
LD_GBLOK  03B05238   03B0AAD0    EXCL                   1          1  03B09378
LD_LBLOK  03B05268                                      0          0
DMLOCKNM  03B05298   03B0B690    EXCL                  35          0
CCSERLCK  03B052C8                                      0          0
==LM: LOCK WAIT QUEUE
LOCK      ADDRESS    -> NEXT     OWNER     MODE  SUSPEND    STATUS
NAME                                           TOKEN
----      -------    -------     -----     ----  -------    ------
LD_GBLOK  03B09378   00000000    03B0B3A0  EXCL  010B0001
```

Figure 8. Lock manager summary information

Table 21 describes each of the fields in the lock manager summary information.

Table 21 (Page 1 of 2). Fields in the lock manager summary information

Field	Description
LOCK NAME	The name given to the lock by the domain that originally issued the ADD_LOCK command.
LOCK TOKEN	The token assigned by the lock manager to uniquely identify the lock.
OWNER	A token that uniquely identifies the owner of the lock. It is blank unless a task currently holds the lock, in which case the KE_TAS number of the task is given.
MODE	The lock mode. It can be: **Blank** No task currently holds the lock. **EXCL** The lock is exclusive—only one task can hold the lock at any one time. The lock owner is identified in the OWNER field. **SHR** The lock is shared—several tasks can hold the lock. In this case, the OWNER field will be blank.
COUNT	Blank unless the lock mode is SHR, when it shows the number of tasks currently holding the shared lock.
# LOCK REQUESTS	The cumulative total of the number of times a lock has been requested—that is, the number of times the LOCK request has been issued for the lock.
# LOCK SUSPENDS	The cumulative total of the number of tasks that have been suspended when requesting this lock because the lock is held by another task.

Table 21 (Page 2 of 2). Fields in the lock manager summary information	
Field	**Description**
-> QUEUE	Blank unless tasks are currently suspended, awaiting the lock. If this is the case, this field contains the address of the first such task. Further information about the task is given in the 'LOCK WAIT QUEUE' section of the information.
ADDRESS	The address of the lock manager LOCK_ELEMENT that represents the suspended task.
-> NEXT	The address of the next task in the queue awaiting the lock. If this field is zeros, this is the last task in the queue.
OWNER	The KE_TAS number of the task that is currently suspended, awaiting the lock.
MODE	The lock mode. It can be: **EXCL** The lock is exclusive—only one task can hold the lock at any one time. The lock requester is identified in the OWNER field. **SHR** The lock is shared—several tasks can hold the lock.
SUSPEND TOKEN	The dispatcher suspend token for the suspended task.
STATUS	The status of the suspended task. It can be: **Blank** The task is waiting to acquire the lock. **DELETED** The suspended task has been deleted from the queue. This occurs only if the lock is deleted. **PURGED** The task was purged while waiting to acquire the lock.

The first step is to establish which lock the suspended task is waiting on. Obtain the KE_TAS number from the dispatcher domain summary for the suspended task and match this with an OWNER in the 'LOCK WAIT QUEUE' section of the lock manager summary information.

In the example, only one task is suspended and waiting to obtain the LD_GBLOK lock. The owner (KE_TAS identifier) of this task is 03B0B3A0.

You then have to find out which task currently holds the lock that the suspended task is waiting on. You can do this by looking at the lock manager summary for that lock—in this case, LD_GBLOK.

If the mode of the lock is SHR (shared), you will not be able to proceed any further and you will have to contact your IBM Support Center.

If the mode is EXCL (exclusive), the identifier of the task that currently holds the lock is given in the OWNER field. In the example, the task that currently has the lock— LD_GBLOK—is 030B0AAD0. Because the OWNER field is the KE_TAS identifier of the task, you can find out from the dispatcher domain summary the status, dispatcher task number, and TCA address of the task that currently holds the lock.

When you have all this information ready, contact the IBM Support Center and report the problem to them.

Journal control waits

Read this section if the resource type your task is waiting on is JASUBTAS or starts with the characters JC, indicating journal control.

Resource type JASUBTAS

The purpose of this wait is to make the shutdown (normal or immediate) wait until the JASP subtask has completely submitted all the archiving jobs of those journals needing to be archived (as determined in the DFHJACD data set).

Resource type JCAVLECB

If the resource type is JCAVLECB and the resource name is LECBECB, the task is waiting for a logical ECB to become available. This type of wait should only occur at times of high journaling activity.

The number of logical ECBs is a function of the number of journals defined in the JCT. Increasing the number of journals increases the number of logical ECBs.

Resource type JCBUFFER

If the resource type is JCBUFFER, with resource name AVAIL_nn ("nn" is a journal number in the range 01 through 99), the buffer is probably nearly full, or the journal is unavailable, for example, because of a volume switch or an output error.

If the resource type is JCBUFFER, with resource name JCTAECB, the task that has requested shutdown is waiting for the journaling task to flush the buffer, close the journal, and terminate itself.

Resource type JCFLBUFF

Tasks are made to wait on resource type JCFLBUFF after they have issued a request to CLOSE a journal, while the buffer is flushed (written to the journal). You can identify the journal for which the close request has been made from the final two characters of the resource name, FLUSH_nn ("nn" is the journal number, in the range 01 through 99).

You are likely to have an extended wait on this resource type only if an I/O problem arises. Check to see if any diagnostic messages have been issued to indicate that this has happened.

Resource type JCIOBLOK

User tasks are suspended on resource type JCIOBLOK, with resource name Jnnbbbbb ("nn" is the journal number, in the range 01 through 99, and "bbbbb" is the block number), when they make a JOURNAL request with the WAIT option. A task suspended in this way with the WAIT option waits until I/O is complete.

An extended wait at this point indicates that there is either an I/O problem or a system error.

Resource type JCIOCOMP

Resource type JCIOCOMP can have any of these resource names associated with it:

- JCTICA
- JCTIOECB
- RDRECDnn
- RDBLOKnn

Only JCTIOECB is associated with waits that can involve user tasks.

Resource name JCTIOECB

User tasks that issue CLOSE requests are made to wait on resource type JCIOCOMP and resource name JCTIOECB while I/O completes. Specifically, the CLOSE request is made to wait for an already active I/O request to complete before proceeding. An extended wait might indicate an I/O problem.

Resource type JCLASTBK

A user task that has requested a journal switch can be made to wait on resource type JCLASTBK and resource name Jnnbbbbb ("nn" is the journal number, from 01 through 99, and "bbbbb" is the block number) when I/O is already active. The wait is for the block identified by "bbbbb" to be written to the journal before it can be closed. It is essentially an I/O wait, and there could be an I/O problem if there is an extended wait at this point.

Resource type JCCLDONE

User tasks can be made to wait on resource type JCCLDONE and resource name SUBTASK, but an extended wait at this point indicates that there is a system error. The wait is for the OPEN/CLOSE subtask to complete the close request, and the user task cannot influence this.

Resource type JCDETACH

A task that has requested shutdown can be made to wait on the detaching of the journal subtask from the operating system.

Resource type JCOPDONE

User tasks can be made to wait on resource type JCOPDONE and resource name SUBTASK, but an extended wait at this point indicates that there is a system error. The wait is for the OPEN/CLOSE subtask to complete the open request, and the user task cannot influence this.

Resource type JCREADY

Resource type JCREADY can have any of these resource names associated with it:

- JCTXAECB
- JCTXBECB

User tasks are made to wait on resource type JCREADY and one of these resource names when no operator reply has been received to message DFH4583, and message DFH4584 has subsequently been issued. For details of the

circumstances under which these messages are issued, and the actions you need to take, see the *VSE/ESA Messages and Codes Volume 3* manual.

The fifth character in the resource name (A or B) identifies the next extent to be used.

Resource type JCRQDONE

User tasks can be made to wait on resource type JCRQDONE and resource name SUBTASK in three routines:

- DFHJCC (user tasks)
- DFHJCO (user tasks)
- DFHJCJOJ (system tasks)

The task is waiting for the journaling subtask to complete an open or close request issued by another task. Extended waits indicate either a system error, or possibly an I/O problem.

Resource type JCSWITCH

If the resource type is JCSWITCH, with resource name DS_SDJnn ("nn" is the journal number, from 01 through 99), a wait is issued on the task that requested shutdown when a journal switch is already active. Extended waits indicate either a system error, or possibly an I/O problem.

Resource type JCTAPE2

Resource type JCTAPE2 can be associated with two different resource names:

- PREOPNnn
- STABLEnn

  ("nn" is the journal number, which is in the range 01 through 99)

The waits are both associated with "open ahead" tape processing, but in different circumstances.

Resource name PREOPNnn

This wait is issued when a volume switch has been requested and "open ahead" processing has not completed. It probably indicates that the volume that is to be opened ahead is not ready.

Resource name STABLEnn

This wait is issued during shutdown processing, when previous "open ahead" processing failed to complete. Like resource name PREOPNnn, it probably indicates that the volume to be opened is not ready.

Journaling cannot be shut down until this processing is complete, and it is potentially a cause of CICS stalling during termination processing.

Task control waits

If your task is waiting on a resource type of KCCOMPAT or KC_ENQ, it has been suspended by the transaction manager. If your task is waiting on a resource type of EKCWAIT, it has been suspended by task control.

KC_ENQ indicates that CICS code acting for a task has issued an EXEC CICS ENQ command or a DFHKC TYPE=ENQ macro. If there is an extended wait for no apparent reason, this might indicate an error within CICS. If that turns out to be the case, contact the IBM Support Center.

EKCWAIT indicates that a task has issued an EXEC CICS WAIT EVENT command. USERWAIT indicates that a task has issued an EXEC CICS WAITCICS or EXEC CICS WAIT EXTERNAL command. If the wait is prolonged, you should identify the event being waited on, and:

- Check that the EXEC CICS command specified the correct event.
- Check for problems with the task that should be completing the work for the specified event. It might be waiting or looping, it might have a performance problem, or it might have failed completely.

If the resource type is EKCWAIT or USERWAIT and the EXEC CICS WAIT EVENT, WAITCICS, or WAIT EXTERNAL command included the NAME option, the specified name is the resource name. For programming information about the NAME option of the WAIT EVENT, WAITCICS, and WAIT EXTERNAL command, see the *CICS Application Programming Reference* manual.

Resource type KCCOMPAT

If you have a resource type of KCCOMPAT, the resource name tells you more about the circumstances of the wait. The meanings of the resource names are described in Table 22.

Table 22. KCCOMPAT waits: meaning of resource names

Resource name	**Meaning**
CICS	The task has been suspended on a DFHKC TYPE=WAIT,DCI=CICS macro call. CICS has issued the macro. The task is waiting for some internal event, and the ECB should be posted by CICS under another task.
LIST	The task has been suspended on a DFHKC TYPE=WAIT,DCI=LIST macro call issued by CICS code. It is waiting for any ECB in a list of ECBs to be posted, after which it is resumed.
SINGLE	The task has been suspended on a DFHKC TYPE=WAIT,DCI=SINGLE macro call issued by CICS code. It is waiting for a single ECB to be posted, after which it is resumed.
TERMINAL	The task has been suspended on a DFHKC TYPE=WAIT,DCI=TERMINAL macro call. CICS has suspended the task. The task is waiting for terminal I/O to complete, and stays suspended until resumed by CICS.

If the resource name for the wait is SINGLE, CICS, or LIST, look at the entry in the SUSPAREA column of the dispatcher summary in the dump. The type of value it contains depends on the resource name:

- For SINGLE or CICS, it is the address of an ECB
- For LIST, it is the address of a list of ECBs

(The contents of the SUSPAREA entry are not significant for TERMINAL, because this type of wait is subject to the dispatcher RESUME function. For more information about debugging terminal waits, see “Terminal waits” on page 94.)

Check the contents of the SUSPAREA entry. Does it contain a valid address? That is, is it within the CICS address space, and actually pointing at an ECB, or a list of ECBs?

If you find an invalid address: It is possible that a storage overlay is the cause of the wait problem. If you suspect this to be the case, turn to Chapter 10, “Dealing with storage violations” on page 173 for further advice. However, note that this is likely to be a “random” overlay, and such problems are often very difficult to solve.

From the kernel information in the dump, find out which code issued the DFHKC macro call. If you think that CICS has passed an incorrect address, contact the IBM Support Center, and report the problem to them.

If you find a valid address: Consider what area the ECB is in. Does the position of the ECB, and its environment, suggest that it relates to a resource whose availability you can control? If so, you might be able to solve the problem by redefining the quantity of that resource.

If the ECB does not lie within an area that you can control, refer the problem to the IBM Support Center.

Resource type KC_ENQ

If your task is waiting on resource type KC_ENQ, it is unconditionally enqueued on a single server resource that is currently unavailable. Typically, tasks are made to wait on KC_ENQ when they make certain types of file control request, if the file is already in use. These are the cases:

- The waiting task has attempted to change the state of a file that is in use. Another task has already attempted to change the state of the same file, and is suspended on resource type FCFSWAIT. For more details, see “Resource type FCFSWAIT—waits for file state changes” on page 110.
- The waiting task has attempted to update a record in a recoverable file while another task has the lock on it. The task owning the record lock retains it until it reaches the end of the current logical unit of work (syncpoint or end of task). For more details of record locking for VSAM files, see “Resource type KC_ENQ—wait for a record lock in a recoverable VSAM file” on page 114.

If the wait on resource type KC_ENQ is prolonged:

- More than one task might be enqueued on the resource, and the task you are investigating could be some way down the list. Check the programming logic of any of your programs accessing the resource, to see if it can be released more quickly. Consider whether you can include EXEC CICS DEQ commands.

- Another (long-running) task might have used the resource and finished with it, without issuing an EXEC CICS DEQ command. The resource is made available automatically when the task terminates, but in the meantime, no other tasks are able to use it.
- There might be a CICS system error. If you have considered the other possibilities and you think this is the most likely explanation, refer the problem to the IBM Support Center.

Storage waits

Read this section if you have found that a task is waiting for a long time on any of the resource types CDSA, UDSA, ECDSA, EUDSA, ERDSA, SDSA, ESDSA, or RDSA. Waits on these resources occur when tasks make unconditional storage requests (SUSPEND=YES) that cannot be satisfied. The type is CDSA, UDSA, SDSA, or RDSA for storage requests below the 16MB line, and ECDSA, EUDSA, ESDSA, or ERDSA for storage requests above the line.

Note that, if conditional requests are made (SUSPEND=NO), tasks are not suspended on these resources. Instead, an exception response is returned if the request cannot be satisfied.

CICS automatically takes steps to relieve storage when it is under stress, for example by releasing storage occupied by programs whose current use count is 0.

In addition, your task may be automatically purged if it has waited for storage longer than the deadlock time-out parameter specified in the installed transaction definition. Certain conditions prevent purging of a task, for example, DTIMEOUT(0) or a specification of SPURGE(NO) on the RDO TRANSACTION resource definition. See the *CICS Resource Definition Guide* for further information on DTIMEOUT and SPURGE.

The two most likely explanations for extended waits on storage requests are:

1. The task has issued an unconditional GETMAIN request for an unreasonably large amount of storage.
2. The task has issued an unconditional GETMAIN request for a reasonable amount of storage, but the system has too little available. It could be approaching SOS, or the storage could have become too fragmented for the request to be satisfied.

The first step is to get a CICS system dump, and format it using the formatting keyword SM. The way you interpret the dump to investigate each of the above possibilities is dealt with in the sections that follow.

Was the request for too much storage?

Look in the SM suspend queue summary. This gives, amongst other things, the number of bytes requested by every task that has been suspended by the storage manager. You can see whether any of them has made a GETMAIN request for an unreasonably large amount of storage. For example, the following is the dump output in the SM suspend queue summary when task 41 requests 10 000 000 bytes:

```
==SM: Suspend queue summary

   KE Task  Tran #  Susptok  Subpool  DSA       Request

   053E5400 0000041 04080011 U0000041 EUDSA    10000016
```

If the suspended task has made a reasonable GETMAIN request, you next need to see if the system is approaching SOS.

Is the storage close to being exhausted?

It could be that your task has made a reasonable request for storage, but the system is too close to SOS for the request to be satisfied.

To see if this could be the cause of the wait, look at the DSA summary in the formatted dump. This tells you the current free space in each DSA, both in bytes and as a percentage of the total storage. It also tells you the size of the largest free area, that is, the biggest piece of contiguous storage. ("Contiguous storage" in this context means storage not fragmented by other records. It is accepted that records too large to fit in a single CI can be split across two or more CIs that are not necessarily contiguous.)

If the largest free area is smaller than the requested storage, this is likely to be the reason why the task cannot obtain its requested storage.

If the amount of free space is unexpectedly small, look at the task subpool summary. If a task has made an unusually large number of GETMAIN requests, this could indicate that it is looping. A looping task might be issuing GETMAIN requests repetitively, each for a reasonable amount of storage, but collectively for a very large amount. If you find evidence for a looping task, turn to Chapter 7, "Dealing with loops" on page 129.

If your task has made a reasonable request and the system seems to have sufficient free storage, you next need to see if fragmentation of free storage is causing the GETMAIN request to fail.

Is fragmentation of free storage causing the GETMAIN request to fail?

If the DSA summary shows that the current free space is significantly greater than the largest free area, it is likely that the DSA has become fragmented.

Temporary storage waits

Read this section if you have found that a user task is waiting on a resource type starting with TS, showing that it is for temporary storage.

Resource type TSAUX

A task is forced to wait on TSAUX if it has made an unconditional request for temporary storage, and the request cannot be met because insufficient auxiliary storage is available. The task has issued an EXEC CICS WRITEQ TS command, with or without the REWRITE option, but without specifying NOSUSPEND and without any code to handle the NOSPACE condition. If SPURGE(YES) is defined for the task on the CEDA DEFINE TRANSACTION command, and a DTIMEOUT

interval other than 0 has been specified, the task is purged when that time expires. Otherwise, it is not purged, and is liable to be suspended indefinitely.

A task that makes a conditional temporary storage EXEC CICS WRITEQ TS request (NOSUSPEND specified) is not suspended if the request cannot be met. Instead, if the required auxiliary storage is not available, an exception response is returned to it. (There might still be a suspension for another reason—for example, the temporary storage program itself might become suspended after issuing a GETMAIN, if CICS went short on storage.)

These are the two most likely reasons why a task that has issued an unconditional EXEC CICS WRITEQ TS request might be suspended on resource type TSAUX:

1. The task has issued a request requiring too large a piece of temporary storage.
2. The task has issued a request requiring a reasonable amount of temporary storage, but there is too little available.

   This could indicate that the amount of auxiliary storage is becoming exhausted. Otherwise, it could be that there is quite a large amount of auxiliary storage left, but the storage is too fragmented for the request to be satisfied.

The first step is to get a CICS system dump, and format it using the formatting keyword TSP to show the temporary storage control blocks. Include formatting keywords SM and KE, too, as you might need to refer to the summaries for these two components as well. Analyze the dump to investigate the cause of the problem as follows:

Was the request for too much temporary storage?

Look in the temporary storage suspend queue summary. This gives, amongst other things, the number of bytes requested by every task that has been suspended because its request for storage cannot be met. You can see whether any of them has made a request requiring an unreasonably large amount of temporary storage.

Is temporary storage close to being exhausted?

It could be that your task has made a reasonable request for temporary storage, but the amount of unallocated space is close to exhaustion.

To see if this could be the cause of the wait, look at the temporary storage summary in the formatted dump. If the current free space is very small, this is likely to be the reason why the task cannot obtain its requested temporary storage. In such a case, consider defining secondary extents for the data set.

Look also at the temporary storage request summary. If a task has made an unusually large number of WRITEQ TS requests, it could be looping. A looping task might be issuing WRITEQ TS requests repetitively, each for a reasonable amount of storage, but collectively for a very large amount. If you find evidence for a looping task, turn to Chapter 7, “Dealing with loops” on page 129.

If your task has made a reasonable request and the system seems to have sufficient unallocated temporary storage, you next need to see if fragmentation of unallocated storage is causing the WRITEQ TS request to fail.

Is fragmentation of unallocated storage causing the WRITEQ TS request to fail?

Look at the temporary storage byte map, TSBMAP, in the dump.

Each byte in the TSBMAP shows how the corresponding control interval (CI) in auxiliary storage is being used. If the CI is completely unused, the byte representing it has a value equal to the number of segments in the CI. The number of segments in each CI is shown in the temporary storage program summary in a system dump. Alternatively, you can find out the number of segments per CI from field TSASPCI in the temporary storage auxiliary control area, TSACA. A value of X'3F', for example, in field TSASPCI would show that each CI has X'3F' segments.

If the CI is completely allocated, you see a value of 0 for it in the TSBMAP. Such a CI cannot be used to satisfy further temporary storage WRITEQ TS requests.

If the TSBMAP shows any value between 0 and the total number of segments in the CI, the CI is partly used. Even if the temporary storage data set is not yet full, that is, one or more of the CIs has unused segments, there must be sufficient contiguous storage available for the WRITEQ TS request before it can be satisfied. You can find out if this is a likely reason for the failure by looking in the TSBMAP at the distribution of unused segments between CIs. The temporary storage program summary in a system dump shows the number of bytes per segment. Alternatively, look in field TSABPSEG of the TSACA, to find this information.

Resource type TSBUFFER

Resource type TSBUFFER indicates that the waiting task has issued an auxiliary temporary storage request, but the buffers are all in use. If you find that tasks are often made to wait on this resource, consider increasing the number of auxiliary temporary storage buffers (system initialization parameter TS).

Resource type TSEXTEND

Resource type TSEXTEND indicates that the waiting task has issued a request to extend the auxiliary temporary storage data set, but some other task has already made the same request. The wait does not extend beyond the time taken for the extend operation to complete. If you have a task that is waiting for a long time on this resource, it is likely that there is a hardware fault or a problem with VSAM.

Resource type TSIO

Resource type TSIO indicates that the task is being made to wait while physical I/O takes place during an auxiliary temporary storage read or write. If there is an extended wait on this resource, it is likely that there is a hardware fault or a problem with VSAM.

Resource type TSQUEUE

Resource type TSQUEUE indicates that the waiting task has issued a request against a temporary storage queue that is already in use by another task. The latter task is said to have the lock on the queue. If the queue is recoverable, the task has the lock until the logical unit of work is complete. If it is not recoverable, the task has the lock for the duration of the temporary storage request only.

If tasks in your system are frequently made to wait on temporary storage queues, consider the following:

- Are tasks that are performing operations on the same temporary storage queue intended to do so, or is the ID of the queue unintentionally not unique?
- Is it possible to create more temporary storage queues to reduce the contention between tasks?
- If the queue in question is recoverable, is it possible to make tasks relinquish control of it more quickly? Consider reducing the size of LUWs, or making conversational tasks pseudoconversational.

Resource type TSSTRING

Resource type TSSTRING indicates that the task is waiting for an auxiliary temporary storage VSAM string. If you find that tasks frequently wait on this resource, consider increasing the number of temporary storage strings (system initialization parameter TS).

Resource type TSUT

If a user task is waiting on resource type TSUT, activity keypointing is taking place. This involves a large amount of I/O, and, if there are many temporary storage queues, it could take a relatively long time to complete.

Resource type TSWBUFFR

Resource type TSWBUFFR indicates that the waiting task has issued an auxiliary temporary storage request, but the write buffers are all in use. You have no control over how temporary storage allocates read buffers and write buffers from the buffer pool, but if you find that tasks are often made to wait on this resource, increasing the number of auxiliary temporary storage buffers (system initialization parameter TS) should help solve the problem.

Terminal waits

Read this section if you have any of the following problems:

- A task should have started at a terminal, but has failed to do so.
- A task is waiting on a resource type of KCCOMPAT, with a resource name of TERMINAL.
- A task is waiting on a resource type of IRLINK, with a resource name of SYSIDNT concatenated with the session name.

Note that, if you have one or more unresponsive terminals (that is, terminals that are showing no new output and accepting no input), this does not necessarily indicate a terminal wait. If you have this problem, use CEMT INQUIRE TERMINAL to find the transaction running at the terminal, and then CEMT INQUIRE TASK to find out what resource that task is waiting on. When you know that, look at Table 20 on page 69 to find where you can get further guidance.

If all the terminals in the network are affected, and CICS has stalled, read “What to do if CICS has stalled” on page 124 for advice about how to investigate the problem.

If yours is a genuine terminal wait, remember when you carry out your investigation that terminals in the CICS environment can have a wide range of characteristics. A terminal is, in fact, anything that can be at the end of a communications line. It could, for example, be:

- A physical device such as a 3270 terminal or a printer
- A batch region
- Another CICS region connected by an interregion communication link
- A system that is connected by an LUTYPE6.1 or APPC protocol.

If LUTYPE6.1 is in use, the other system might be another CICS region, or IMS. With APPC, the range of possibilities is much wider. It could include any of the systems and devices that support this communications protocol. For example, apart from another CICS region, there might be a PC system at the other end of the link.

If you find that the fault lies with a terminal, find out what type it is. In some cases, you probably need to look in appropriate books from other libraries for guidance about problem determination.

Terminal waits—first considerations

Here are a few preliminary considerations that could point to a simple solution.

- Is there an obvious physical explanation for the wait? For example, is a terminal operator failing to respond to a request for input? In the case of a printer, has it been powered off, or run out of paper?
- Have you checked in the CSTL and CSNE logs to see if there is a message you might have missed? If either DFHTCP or DFHZCP detected an error related to terminal control, a message reporting the problem will have been sent to the CSNE log, and, perhaps, to the console too.

  If you do find a message there reporting some terminal error that can be related to the task, it should give you an idea of why the task is waiting. You can find an explanation of the message and a description of the system action in response to the error by looking in the *VSE/ESA Messages and Codes Volume 3* manual.

  The CSNE log entry may also show that an error was detected, but that no action was taken by TACP or NACP. This can occur if the line or terminal was out of service, or if the error actions had been turned off in the user exits of DFHTEP and DFHNEP. In such a case, the CICS code enabling the waiting task to be resumed is never executed, and the task waits indefinitely.
- Have any HANDLE CONDITION routines for terminal errors been coded incorrectly? If an attempt were made, in such a routine, to access the terminal that has the error, the application would be very likely to wait indefinitely.
- If the terminal is installed using autoinstall, has the system failed to load DFHZATA, the autoinstall program, or DFHZCQ which is called by DFHZATA, because of a short on storage condition? If so, you need to deal with the cause of the short on storage condition. Check the delete delay that you have specified—if it is too short, your system may be deleting and reinstalling terminals unnecessarily. If storage fragmentation is preventing DFHZATA or DFHZCQ from being loaded, consider defining them as resident (RESIDENT(YES) on the RDO PROGRAM definition). However, be aware that

it is a large program, and check your storage requirements before making this change.

If none of these considerations applies, start a systematic investigation into the reason for the wait, as follows:

Terminal waits—a systematic approach

First determine the type of terminal involved in the wait, and the type of access method in use for that terminal. Both of these factors influence the way you perform problem determination.

Then find where in the communication process the fault lies:

1. Is the problem associated with the access method?
2. If the access method has returned, or has not been involved, is terminal control at fault?
3. If terminal control is working correctly, why is the terminal not responding?

To answer most of these questions, you will need to do some offline dump analysis. Use CEMT PERFORM SNAP to get the dump, and then format it using the formatting keyword TCP.

Note: Do not cancel your task before taking the dump. If you do, the values in the terminal control data areas will not relate to the error.

What type of terminal is not responding?

You can check the terminal type in the following ways:

Online method: Use the transaction CECI to execute the EXEC CICS INQUIRE TERMINAL DEVICE system programming command. This returns one of the terminal types identified in the *CICS Resource Definition Guide*.

Offline method: Look at the formatted dump output you have obtained for keyword TCP. First, identify the TCTTE relating to the terminal, from the four-character terminal ID shown in field TCTTETI. Now look in field TCTTETT, and read the 1-byte character that identifies the type of terminal. You can find what terminal type is represented by the value in the field from the description given in the *CICS Data Areas* manual.

What type of access method is in use?

Find the type of access method being used by the terminal that is not responding:

Online method: Use the CECI transaction to execute the EXEC CICS INQUIRE TERMINAL ACCESSMETHOD system programming command. This returns the access method in use by the terminal.

Offline method: You can find the access method for the terminal from the TCTTE. Look in field TCTEAMIB, which is the byte name definition for the access method. The *CICS Data Areas* manual relates values to access methods.

The most common access method is VTAM, identified by a value of TCTEVTAM. The problem determination procedures outlined focus exclusively on VTAM. You might also find either of the following values, each being of special significance for problem determination:

TCTEISMM The access method is MRO. This is used for interregion communication, and it shows that the resource not responding is a remote CICS region. The most likely reason for the wait is that some task in the remote region is also in a wait state. The way you deal with this type of problem is described in “Your task is waiting on another CICS region” on page 101.

TCTELU6 Intersystem communication (ISC). The resource that is not responding is a remote system. If this is a CICS system, diagnose the problem in the remote system using the techniques given in this book. If the remote system is a non-CICS system, you might need to refer to documentation provided for that system.

VTAM in use—debugging procedures

First, look in the CSNE log to see if there is an error message that explains the wait. If it contains an error code, you can find out what it means from section “VTAM codes” in the *VSE/ESA Messages and Codes, Volume 2* manual.

Look next for any NACP error codes in fields TCTEVRC5, TCTEVRC6, TCTEVRC7, and TCTEVRC8 of the terminal table entry, TCTTE. Look also for any SNA sense code in field TCTEVNSS.

Is the problem associated with VTAM?

You can find the VTAM process status (with respect to the waiting task) from fields TCTEICIP and TCTEIDIP in the TCTTE. You might find the following values there, and their interpretations:

TCTECIP Command request in progress

TCTEDIP Data request in progress

Either of these status values indicates that a VTAM request is in progress, and that the VTAM RPL is active. A response is expected either from VTAM, or from the terminal. You can find the address of the RPL from field TCTERPLA, unless the request was for a RECEIVE on an APPC session, in which case you can find the RPL address from field TCTERPLB.

If a VTAM request is not in progress, the next field of interest is in the VTAM system area of the TCTTE. Find four bytes of VTAM exit IDs, starting at field TCTEEIDA. If any are nonzero, the VTAM request has completed. Nonzero values suggest that VTAM is not involved in the wait. You can find the meanings of the values from the VTAM module ID codes list in the *CICS User’s Handbook*.

If you suspect that the problem is associated with VTAM, consider using VTAM buffer tracing. This technique can give you detailed information about the execution of VTAM requests. For guidance about using the technique, read the appropriate section in Chapter 14, “Using traces in problem determination” on page 199.

VTAM in use—is terminal control at fault?

First look at field TCTVAA1 in the terminal control table prefix, TCTFX. This contains either the address of the first TCTTE on the active chain, or the value X'FFFFFFFF'. If you see the latter value, it means that terminal control does not recognize that it has work to do. If this conflicts with the INQUIRE TASK report you have received that the task is waiting on some terminal related activity, report the problem to your IBM Support Center.

If field TCTVAA1 points to a TCTTE on the active chain, check that the TCTTE of the terminal your task is waiting for is included in the chain. You can find this out by following the chain using the “next” pointer, field TCTEHACP of the TCTTE. If it does not contain the address of the next TCTTE on the chain, it contains one of these values:

Value	Meaning
X'FFFFFFFF'	This is the last TCTTE on the chain
X'00000000'	This TCTTE is not on the active chain

If you find a value of X'00000000', report the problem to the IBM Support Center.

VTAM in use—is the terminal at fault?

If you have found that the access method and terminal control are not causing the wait, the terminal itself must be waiting for some reason. Look at fields in the TCTTE for the terminal to find its status.

CICS system dumps contain an index to the VTAM terminal entries. It appears in the terminal control (TCP) summary section of the dump.

Information about the status and attributes of the VTAM terminals appears in an interpreted form at the end of the control block for each terminal entry. The information shown depends on the attributes of the terminal.

The example in Figure 9 on page 99 shows the index followed by a terminal entry with its interpreted status and attribute information.

```
===TCP: TERMINAL CONTROL SUMMARY (VTAM TERMINALS)
        TERMINAL                 TASK         IN        ERROR    ACTIVE RPL   WORK    ZNAC    INTERVENTION   AUTOINSTALL
TERMID    TYPE     LOGGED ON    ATTACHED     SERVICE     STATS.    REQUEST     TO DO  QUEUED     REQUIRED       ACTIVITY
 R51       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 R52       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 R53       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 R54       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 R55       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 S51       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 S52       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 S53       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 S54       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 S55       C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 -AAA      C0         NO           NO          YES      00000001      NO         NO      NO          NO            N/A
 -AAB      C0         NO           NO          YES      00000000      NO         NO      NO          NO            N/A
 TCTTE.R51 03B5C420 TCT TERMINAL ENTRY
    0000  D9F5F140 C0000504 03B5C424 00000000  00000000 00000000 00000000 00080000  *R51 ......D.....................*    03B5C420
    0020  00000000 0C000000 C5D5E400 00008080  00000000 00000000 00000000 00000000  *........ENU.....................*    03B5C440
    0040  00000000 00000000 00000000 00000000  00000000 01D80000 00000000 03B22030  *.....................Q..........*    03B5C460
    0060  00000000 00000000 00000000 03B58690  00000000 00000000 03B46390 00000000  *..............f................*    03B5C480
    0080  00000000 00000000 00000000 00000000  03B430F8 00000000 00000000 00000000  *...................8............*    03B5C4A0
    00A0  00000000 00000000 00000000 00840000  00000000 00000000 00000000 00000000  *.............d................*    03B5C4C0
    00C0  00000000 80000000 00000000 00000000  00000000 0000003B 01000000 00000000  *................................*    03B5C4E0
    00E0  10000000 00000000 00000000 03B5C618  00000000 00000000 00000000 00000000  *..............F................*    03B5C500
    0100  3A008400 00000000 00000000 00000000  00000000 FFFF0000 00000000 00000000  *..d............................*    03B5C520
    0120  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B5C540
    0140  10001000 10000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B5C560
    0160  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*    03B5C580
    0180  08090000 00000000 00000000 00000000  00000000 00000000 00000000 03B59421  *............................m.*    03B5C5A0
    01A0  00440000 00001008 D4FD6038 80800014  00000000 00000000 00000000 00000000  *........M.-.....................*    03B5C5C0
    01C0  00000000 00000000 00000000 00000000  00000000 00000000                    *........................       *    03B5C5E0
    TERMID = R51                           EXIT FOOTPRINTS (HEX) = 00000000
    IN SERVICE                             TCTTECA (NO TASK ATTACHED)
    TCTECCV (STARTED BY TTI)               TCTECSM (CA-MODE)
    INPUT STATISTICS (DECIMAL) = 00000000  OUTPUT STATISTICS (DECIMAL) = 00000000
    ERROR STATISTICS (DECIMAL) = 00000000  TCTE1RY (CICS IS PRIMARY)
    TCTELSE (LUC CONTENTION LOSER)         TCTESBIF (SBI/BIS SUPPORTED)
```

Figure 9. Terminal index and terminal entry with interpreted information

The values that are given below for fields in the TCTTE are not the only possibilities, but they show important things about the terminal status. If you find any other values for these fields, look in the *CICS Data Areas* manual to find out what they mean.

Ask the following questions. The values shown indicate possible solutions.

1. Is the terminal in service? Look at field TCTTETS of the TCTTE, to find the terminal status. The values that indicate why a terminal was failing to respond include:

   ```
   TCTTESPO = 1 and TCTTESOS = 1     terminal out of service
   TCTTESOS = 1 only                 terminal in error recovery
   ```

   Look also at field TCTESEST, to find the session status with respect to automatic transaction initiation (ATI) for the terminal. Possible values include:

   ```
   TCTESLGI = 0     CREATESESS(NO) in TYPETERM definition
   TCTESLGI = 1     CREATESESS(YES) in TYPETERM definition
   TCTESLGT = 1     recovering CREATESESS
   ```

   A value of TCTESLGI = 0, with TCTESLGT = 0, too, shows that CREATESESS(NO) has been specified in the RDO TYPETERM resource definition. This means that EXEC CICS START requests and ATI requests cannot cause a session to be created. The request is either queued or rejected when no session is currently established. This can put a terminal into an indefinite wait state, especially if the terminal is a printer.

A value of TCTESLGI = 1 shows that CREATESESS(YES) has been specified in the RDO TYPETERM resource definition. This means that CICS is allowed to create sessions for the terminal, so the CREATESESS status is not the cause of the wait.

A value of TCTESLGT=1 means that the session is in error recovery. This could explain why there is no response from the terminal.

2. Has a task been created for this terminal? Look first at field TCTTECA of the TCTTE.

   - If its value is nonzero, there is a task attached to the terminal. Field TCTEICCV indicates the following:

     ```
     TCTECCV = 0      the task has been started by a terminal
     TCTECCV = 1      the task has been started by ATI
     ```

   - If its value is zero, look also in fields TCTTEIC and TCTECTI. The values you might find there are:

     ```
     TCTEOIC = 1      ATI is waiting to start
     TCTECTI = 1      there is ATI work for ZCP to do
     ```

3. Is there a task related to the terminal? From field TCTEMOST, you can find the task session state with VTAM, and also, if bracket protocol is required (from field TCTEIBPE), its conversation state with the terminal from field TCTEIINB. The significant values that might provide further clues to the cause of the wait are:

   ```
   TCTECSM = 1      the task is in conversation with the terminal
   TCTECSM = 0      terminal control will accept new tasks from
                    the terminal
   ```

   Look now at field TCTEIBPE, to see if bracket protocol is required: A value of 1 indicates that bracket protocol is required. If this is the case, look in field TCTEIINB which can have the following values:

   ```
   TCTEINB = 0      a conversation has not been started
   TCTEINB = 1      a task is in conversation with the terminal
   ```

4. Is the terminal logged on to CICS? Look first at the node session status in the TCTTE. The three stages of session creation are represented by three separate bits, in fields TCTEILOS, TCTEIOPD, and TCTEINSD:

   ```
   TCTELOS = 1      the node is logged on
   TCTEOPD = 1      VTAM OPNDST macro issued
   TCTENSD = 1      Start Data Traffic sent
   ```

   If all three bits are set, so the value of the byte is TCTENIS, the node is in session.

   Now see if the terminal is logging off, or if it has already been logged off. The fields of interest are TCTEINND, TCTEINBD, and TCTEIPSA. The values to look for are:

   ```
   TCTENND = 1      the terminal is to be logged off
   TCTENBD = 1      the terminal is logging off because of an error
   TCTEPSA = 1      the session with the terminal ended abnormally
                   —look for any explanatory message on CSMT
   ```

   If any of these bits are set, the terminal might not be able to respond to the waiting task.

5. Should the terminal respond to the task? A value of 1 in field TCTEIPRA tells you that it should.

If the values you find in all these fields suggest that the terminal status is normal, the terminal is probably waiting for some activity to complete before it responds. The type of investigation you need to do next depends on the type of terminal involved in the wait. You should already have determined this, for example by using the EXEC CICS INQUIRE TERMINAL DEVICE system programming command. See the *CICS System Programming Reference* for further information about this command.

Tools you can use for debugging terminal waits when VTAM is in use

The VTAM buffer trace is likely to be of particular use for investigating terminal waits in a VTAM environment. This is a feature of VTAM itself, and you should read the appropriate manual in the VTAM library for details of how to use it.

For a description of the use of this type of tracing in CICS problem determination, see Chapter 14, “Using traces in problem determination” on page 199.

Your task is waiting on a physical terminal

If your task is waiting on a physical terminal, the terminal should first be checked physically to see why it is not responding. If the terminal is at a remote location, ask someone else to check it for you. Some possibilities are:

- A terminal with a keyboard might be waiting for an operator to enter some data.
- A printer might have been powered off, or it could have run out of paper.

Consider also the possibility of hardware error in the terminal itself.

Your task is waiting on another CICS region

If a session has been acquired and has not failed, your task is likely to be waiting for some response from a task in the other region. This can apply to any of the interregion or intersystem communication activities—function shipping, asynchronous processing, transaction routing, distributed transaction processing, or distributed program link. No matter which of these applies, it is most likely that the other CICS region is not responding because the related task there has been suspended.

You need to identify the related task in the remote CICS region, and find out the resource it is waiting on. When you have done that, see Table 20 on page 69 to find out which part of this section to turn to next.

Investigating the related task in the remote region

First, identify the region that is not responding to your local task.

If the task is using interregion communication (IRC), look first at the name of the resource being waited on, returned together with resource type IRLINK by CEMT INQUIRE TASK. The first four characters give you the SYSIDNT of the remote CICS region.

If the task is using intersystem communication (ISC), look in field TCTTEIST of the TCTTE, which points to the ISC system table entry. The first field in the system table entry is the identity of the remote CICS region.

When you have identified the CICS region, take a system dump of it. Either use the CEMT PERFORM DUMP command in that CICS region, or use the VSE DUMP command. Take a system dump of the local CICS region, too.

Format the dumps using the formatting keyword DS to get a summary of the tasks in each region, and TCP to get the TCTTEs.

First find the TCTTE for the task in the local CICS region. The way you find the TCTTE for the task in the remote CICS region depends on whether you are using LUTYPE6.1 sessions (or IRC) or APPC sessions:

- For LUTYPE6.1 sessions and IRC sessions, look in local control block TCTENIB, the TCTTE extension for the NIB descriptor, at field TCTESQP. This gives you the session qualifier pair for the session. It provides the terminal ID associated with the local task, concatenated with the terminal ID associated with the remote task.

  Now go to the dump of the remote CICS region, and use the terminal ID to locate its TCTTE. Check in field TCTESQP of TCTENIB to make sure that the session qualifier pair matches that in the local system. It should be made up of the same terminal IDs, but with their order reversed.

- For APPC sessions, look in local control block TCTTELUC, the APPC extension, in field TCTESII. Ignoring the high-order byte, this gives you the session instance identifier of the session.

  Now go to the dump of the remote CICS region, and use the session instance identifier you have found for the remote task to locate its TCTTELUC. The TCTTE precedes the TCTTELUC in the dump.

When you have confirmed that you have located the correct TCTTE, look in field TCTTECA. This gives you the TCA address of the task that is not responding.

Using the TCA address as the entry point, you can now investigate the reason why the task has not responded. It is very likely that it has been suspended because some resource is unavailable. Look in the dispatcher and transaction manager summaries. If you can identify your task, you can see what resource it is waiting on.

When you have identified the resource, turn to the appropriate section in this chapter for guidance about investigating waits on that resource.

VTAM terminal control waits

VTAM terminal control waits are associated with the following resource types:

- ZC
- ZC_ZCGRP
- ZC_ZGRP
- ZC_ZGUB
- ZCIOWAIT
- ZCZGET
- ZCZNAC
- ZXQOWAIT
- ZXSTWAIT

The implication of waits on any of these VTAM terminal resource types are described as follows:

Resource type ZC

If your task is waiting on a resource name of DFHZCRQ1, it is waiting for I/O to complete. The task is attempting to complete one of the following:

- RESETSR
- A send synchronous data flow
- A send asynchronous command
- SESSIONC.

The task waits for the time specified in the RTIMEOUT value of the profile used by the transaction rounded up to the nearest multiple of 16.78 seconds. If the task times out, it receives either an AKCT or AZCT abend.

If your task is waiting on a resource name of DFHZEMW1, the error message writer module, DFHZEMQ, is waiting for the completion of I/O. If a timeout value exists and is exceeded, the suspend expires.

If your task is waiting on a resource name of DFHZRAQ1, this means a READ has been issued. The task is resumed once the I/O operation is complete. If a timeout value exists and is exceeded, the suspend expires.

If your task is waiting on a resource name of DFHZRAR1, this means a READ has been issued. The task is resumed once the I/O operation is complete. If a timeout value exists and is exceeded, the suspend expires.

Resource type ZC_ZCGRP, resource name ZSLSECB

DFHZSLS has to set the TCT prefix VTAM fields from the ACB. This wait is issued to ensure that these fields are set before being used. The wait expires after 3 minutes.

Resource type ZC_ZGRP, resource name PSINQECB

If the task is waiting on resource name PSINQECB, this means that DFHZGRP has issued the VTAM INQUIRE PERSESS macro during persistent session restart, or during the reopening of the VTAM ACB, and is waiting for a response from VTAM. The wait expires after 8 minutes if VTAM does not respond.

Resource type ZC_ZGRP, resource name PSOP2ECB

If the task is waiting on resource name PSOP2ECB, this means that DFHZGRP has issued the VTAM OPNDST RESTORE macro during emergency restart, or during the reopening of the VTAM ACB, and is waiting for a response from VTAM. The wait expires after 8 minutes if VTAM does not respond.

Resource type ZC_ZGRP, resource name TCTVCECB

The persistent sessions initialization module, DFHZGRP, waits for the posting of TCTVCECB to ensure that DFHTCRP has finished installing TCTTEs, before the CNOS value is reinstated for all LU6.2 connections.

Resource type ZC_ZGUB

DFHZGUB issues ten VTAM CLSDST or TERMSESS macros during persistent sessions restart. It waits for an RPL to become free for VTAM to post the VTAM exit. The wait expires after 5 minutes if VTAM does not respond.

Resource type ZCIOWAIT

Suspends on resource type ZCIOWAIT occur when the task is waiting for some terminal I/O. Once the expected I/O event occurs, the task is resumed.

Resource type ZCZGET

If your task is waiting on a resource name of DFHZARL2, it is suspended by module DFHZARL, which deals with application request logic for APPC devices. The suspend is caused by a GETMAIN call to DFHZGET failing. DFHZGET is continually invoked until the GETMAIN is successful.

Resource type ZCZNAC

Suspends on resource type ZCZNAC are on resource names DFHZARL3 or DFHZERH4. The wait is for DFHZNAC to issue an error message. The error message to be issued depends on the error that led to the suspend. Various actions may be taken by DFHZNAC before control is returned to the suspended task.

Resource type ZXQOWAIT

The XRF queue organizer, DFHZXQO, waits for the posting of TCAICTEC and XQOVECTE which happens when the queue is emptied.

Resource type ZXSTWAIT

The XRF session tracker, DFHZXST, waits for the posting of TCAICTEC and TCTVXPLE which happens when the session tracking queue is emptied.

Interregion and intersystem communication waits

If you have a user task that is waiting for resource type ALLOCATE, it has attempted to get a session with another CICS region, but all the sessions are in use. Consider defining a greater number of sessions, which should solve the problem. For guidance about this, see the *CICS Resource Definition Guide* and the *CICS Intercommunication Guide*.

If you otherwise have a problem that you have identified as an interregion or an intersystem communication wait, investigate it as for terminal waits. This is dealt with in “Terminal waits” on page 94.

The method of debugging is the same in each case. You need to consider the access method, terminal control, and the “terminal” itself.

For interregion and intersystem communication, the remote CICS region or system *is* the terminal. Its status can be found using the same online or offline techniques that you would use to find the status of a physical terminal. The status may lead you to suspect that the task running in the remote CICS region is the cause of the problem, and you then need to investigate why that task is waiting. So you could

find that what started as a terminal wait might, after all, be a wait on some other type of resource.

Transient data waits

Tasks issuing requests to read and write to transient data destinations can be made to wait for several different reasons. The reasons depend on the type of request being made, and whether the task is attempting to access an extrapartition or an intrapartition queue.

The resource types that might be associated with the wait are:

- TD_INIT
- TDEPLOCK
- TDIPLOCK
- KC_ENQ
- Any_MBCB
- Any_MRCB
- MBCB_xxx
- MRCB_xxx

The resource name is the transient data queue name, except in the case of TD_INIT, whose resource name is DCT.

Resource type TD_INIT—waits during initialization processing

A second stage PLT program being executed during system initialization can issue a request for a resource that is not yet available, because the component that services the request has not yet been initialized. If the program issues a transient data request that cannot be serviced, it is suspended on a resource type of TD_INIT with a resource name of DCT.

You are unlikely to see any evidence for this type of wait, unless you have trace running during initialization with DS level-1 tracing selected. An error at this stage would be likely to cause CICS to stall (see “CICS has stalled during initialization” on page 124), or to terminate abnormally.

Resource type TDEPLOCK—waits for transient data extrapartition requests

If you have a task suspended on resource type TDEPLOCK, with a resource name corresponding to a transient data queue name, the task has issued a request against an extrapartition transient data queue. Another task is already accessing the same queue, and the waiting task cannot resume until that activity is complete.

If the wait is prolonged, it could be for either of these reasons:

- It is necessary for a task to change VSE subtask mode to open and close a data set. The task must relinquish control while this happens, and, depending on the system loading, this might take several seconds. This contributes to the wait that the second task, suspended on resource type TDEPLOCK, experiences.
- CICS uses the access method SAM to write data to extrapartition transient data destinations. SAM executes synchronously with tasks requesting its services. This means that any task invoking a SAM service must wait until the SAM

processing is complete. If, for any reason, SAM enters an extended wait, the requesting task also experiences an extended wait.

If tasks frequently get suspended on resource type TDEPLOCK, you need to determine which other transactions write data to the same extrapartition destination. You might then consider redefining the extrapartition destinations in the DCT (destination control table).

Resource types TDIPLOCK, KC_ENQ, Any_MBCB, Any_MRCB, MBCB_xxx, and MRCB_xxx

If your task is waiting on any of the resource types TDIPLOCK, KC_ENQ, Any_MBCB, Any_MRCB, MBCB_xxx, or MRCB_xxx, it has made a transient data intrapartition request that cannot be serviced at once. In each case, the resource name identifies the intrapartition queue that the request has been issued against.

Resource type TDIPLOCK

If you have a task suspended on resource type TDIPLOCK, with a resource name corresponding to a transient data queue name, the task has issued a request against an intrapartition transient data queue. Another task is already accessing the same queue, and the waiting task cannot resume until that activity is complete.

If tasks frequently get suspended on resource type TDIPLOCK, you need to determine which other transactions use the same intrapartition destination. You might then consider redefining the intrapartition destinations in the DCT.

You can find further guidance information about the constraints that apply to tasks writing to intrapartition destinations in the *CICS Application Programming Guide*. For more details of the properties of recoverable transient data queues, see the *CICS Resource Definition Guide*.

Resource type KC_ENQ

If a transient data queue has been defined in the DCT as intrapartition and logically recoverable, there are further restrictions (in addition to those leading to waits on resource type TDIPLOCK) on the use of the queue by more than one task at a time.

If you have a task suspended on resource type KC_ENQ, it may have issued a request against such a queue. Be aware, however, that KC_ENQ is not exclusively for transient data waits.

Changes to logically recoverable queues are not committed until the end of a logical unit of work. To ensure integrity of data, therefore, other tasks may be forced to wait until the task accessing the queue has reached syncpoint or the end of its activity. Tasks suspended on resource type KC_ENQ, where the current task is trying to access transient data, are very likely to be waiting for the task currently accessing the queue to reach syncpoint or end of task.

There are two types of KC_ENQ wait: one for reads and one for writes. A delete requires both the read and write lock. No task may, therefore, read or write to a queue while a delete is in progress. A delete cannot proceed until any task currently reading or writing has completed its activity.

A task may read while another task is writing but it can only read already committed records. Uncommitted records can only be read by the task that wrote them.

Whether or not a read request waits on resource type KC_ENQ depends on the coding of the NOSUSPEND option on the READQ command. If coded, QBUSY is returned to the application and the task does not wait. If not coded, the task waits on resource type KC_ENQ.

Resource type Any_MBCB

If your task is waiting on resource type Any_MBCB, the resource name is the name of an intrapartition queue that it is attempting to access. This type of wait shows that all the transient data I/O buffers are in use, and the task resumes only when one becomes available.

Tasks are only likely to wait in this way in a heavily loaded system.

Resource type Any_MRCB

When a transient data I/O buffer has been acquired for a task, a VSAM string must be obtained. If all the VSAM strings available for transient data processing are in use, the task is suspended on resource type Any_MRCB, with a resource name equal to the intrapartition queue name.

Waits on Any_MRCB should not be prolonged, except in a heavily loaded system.

Resource type MRCB_xxx

A resource type of MRCB_xxx, with a resource name equal to an intrapartition transient data queue name, shows that the suspended task has successfully obtained a VSAM string, and is now waiting for VSAM I/O to complete. This should not be a long wait, unless operator intervention is required.

Resource type MBCB_xxx

If a task is waiting on resource type MBCB_xxx, with a resource name equal to the intrapartition queue name, this indicates contention for a transient data I/O buffer. It should not be an extended wait, although it is dependent on VSAM I/O taking place on behalf of another task that has issued a transient data request. If that, for any reason, takes a long time, the wait on resource type MBCB_xxx is correspondingly long. (For descriptions of the waits that might occur during transient data VSAM I/O processing, see “Resource type Any_MRCB” and “Resource type MRCB_xxx”).

The reason for this type of wait is best illustrated by example, as follows:

1. Task #1 issues a transient data request that requires access to an intrapartition queue. Before the request can be serviced, task #1 must be assigned a transient data I/O buffer that is not currently being used by any other task.

   I/O buffers each contain a copy of a control interval (CI) from a data set. Each CI contains records that correspond to elements in an intrapartition queue. A search is made to see if the CI required for task #1 is already in one of the I/O buffers. If it is, that I/O buffer can be used to service the request made by task #1, and no VSAM I/O is involved. If it is not, task #1 is allocated any buffer, so the required CI can be read in. The current contents of the buffer is overwritten.

An I/O buffer can have a R/O (read only) status or a R/W (read/write) status. If the buffer that is allocated to task #1 has R/W status, it contains a copy of a CI that has been updated by some other task, but not yet written back to the data set. Before the buffer can be used by task #1, the CI it contains must be preserved by writing it back to the data set.

2. A request now arrives from task #2, and the request requires the CI that is currently being written to the data set. No two buffers can contain the same CI, so task #2 is made to wait on resource type MRCB_xxx until the outcome of the VSAM I/O is known.

   If VSAM I/O was successful, task #2 is resumed and assigned some other I/O buffer.

   If VSAM I/O was unsuccessful, task #2 can use the I/O buffer that already contains the CI it needs.

Loader waits

A task is suspended by the loader domain if it has requested a program load and another task is already loading that program. Once the load in progress is complete, the suspended task is resumed very quickly and the wait is unlikely to be detected.

Note that the loader does not suspend a task while a program is loaded if it is the first one to ask for that program.

If the requested program is not loaded quickly, the reasons for the wait need to be investigated. The possible reasons for the wait, and the ways you should investigate them are:

1. The system could be short on storage (SOS), so only system tasks can be dispatched.

   To check if the system is short on storage use the CEMT transaction to issue the command CEMT INQUIRE SYSSTEM SOSSTATUS. Examine the job log, check the run statistics, or issue a CEMT INQUIRE DSAS command. If SOS has been reached too often, take steps to relieve the storage constraints. For guidance about this, see the *CICS Performance Guide*.

2. There could be an I/O error on a library.

   Check for messages that might indicate this. If you find one, investigate the reason why the I/O error occurred.

3. There could be an error within VSE.

   Has there been any sort of message to indicate this? If so, you should refer the problem to the IBM Support Center.

File control waits

Most file control waits are associated with resource types starting with the characters FC. Some are associated with resource type KC_ENQ, but KC_ENQ is not used exclusively for file control waits.

Here is a list of identifiable resource types associated with file control waits, and all the possible reasons for waits:

- **FCBFWAIT**
  The task is waiting for a VSAM buffer.
- **FCCIWAIT**
  The task is waiting for VSAM control interval split processing to complete.
- **FCDWWAIT**
  The task is waiting for VSAM upgrade set activity to complete.
- **FCFSWAIT**
  The task is waiting to change the state of a file.
- **FCIOWAIT**
  The task is waiting for I/O on a disk volume.
- **FCPSWAIT**
  The task is waiting for a private string.
- **FCRBWAIT**
  The task is waiting because file recovery failed to complete.
- **FCSRSUSP**
  The task is waiting for a shared resource string.
- **FCTISUSP**
  The task is waiting for a VSAM transaction ID.
- **FCXCWAIT**
  The task is waiting for exclusive control of a VSAM control interval.
- **KC_ENQ**
  The task is waiting for a record lock in a recoverable VSAM file.
- **KC_ENQ**
  The task is waiting for exclusive control of a record in a DAM file.

The implications of waits on any of these file control resource types are dealt with in the sections that follow.

Resource type FCBFWAIT—waits for VSAM buffers

If your task is waiting on resource type FCBFWAIT, it means that a VSAM buffer is not currently available. You can specify the number of VSAM data buffers and VSAM index buffers in the RDO FILE resource definition using the DATABUFFERS and INDEXBUFFERS parameters, respectively.

Consider increasing the numbers of these buffers if you find that tasks are frequently having to wait on this resource type.

If there are insufficient data and index buffers for a single task, the task is suspended indefinitely. This might happen unexpectedly if you have a base cluster and one or more paths in the upgrade set, and your application references only the base. VSAM upgrades the paths whenever changes are made to the base. There could then be too few buffers defined in the RDO LSRPOOL for both base and paths.

Resource type FCCIWAIT—wait for VSAM control interval split activity

If your task is waiting on resource type FCCIWAIT, it means that the task is waiting **within** VSAM for control interval split processing for the file to complete.

Resource type FCDWWAIT—wait for VSAM upgrade set activity

If your task is waiting on resource type FCDWWAIT, it has received a VSAM response, which might indicate that your task is trying to read a record via a VSAM path while this record is being updated by another request. This other request is updating the record either via the base or via another path. If VSAM has not yet completed the update, the content of the alternate index currently in use is no longer the same as the content of the base data set.

This is a transient condition. CICS waits for all current update operations for this VSAM data set to complete and retries the request once only. If the error continues after the request is retried, CICS assumes that there is a genuine error and returns a response of ILLOGIC to the application. Since ILLOGIC is a response to all unexpected VSAM errors, CICS also returns the VSAM response and reason codes (X'0890') in bytes 2 and 3 of EIBRCODE. These identify the cause of the ILLOGIC response.

Resource type FCFSWAIT—waits for file state changes

If your task is waiting on resource type FCFSWAIT, it means that it has attempted to change the state of a file—for example, to CLOSE or DISABLE it—but another task is still using the file. This can happen if a long-running transaction, possibly conversational, is using a recoverable file. The long-running transaction has the lock on the file, and might not release it until it terminates. In such a case, consider changing the programming logic so that the file is released more quickly.

Only one task at a time waits on FCFSWAIT. If any other tasks attempt to change the state of the same file, they are suspended on resource type KC_ENQ. See “Task control waits” on page 88.

Resource type FCIOWAIT—wait for VSAM I/O

If you have a task waiting on resource type FCIOWAIT, it means that the task is waiting **within** VSAM for I/O to take place.

A wait on resource type FCIOWAIT occurs when the exclusive control conflict is deferred internally by VSAM and not returned as an error condition to CICS. An example of this is when a request against an LSR file is made for exclusive control of a control interval (for example, by EXEC CICS WRITE or READ UPDATE) and either this task or another task already holds shared control of this control interval (for example, by STARTBR).

Exclusive control waits are discussed further in “Resource type FCXCWAIT—VSAM exclusive control wait” on page 112.

Resource types FCPSWAIT and FCSRSUSP—waits for VSAM strings

If your task is waiting on either of resource types FCPSWAIT or FCSRSUSP, it means that it cannot get a VSAM string. FCPSWAIT shows that the wait is for a private string, and FCSRSUSP shows that the wait is for a shared resource string. You can purge the task from the system, if the task is purgeable.

The number of strings defined for a VSAM data set (STRINGS parameter in the RDO FILE resource definition) determines how many tasks can use the data set concurrently. STRINGS can have a value in the range 1–255. When all the strings are in use, any other task wanting to access the data set must wait until a string has been released.

If tasks are being caused to wait unduly for strings, consider whether you can increase the value of STRINGS, or change the programming logic so that strings are released more quickly.

An example of programming logic that can hold onto strings (and other VSAM resources) for too long is when a conversational transaction issues a EXEC CICS STARTBR or READNEXT and then enters a wait for terminal input without issuing an EXEC CICS ENDBR. The browse remains active until the ENDBR, and the VSAM strings and buffers are retained over the terminal wait. Also, for an LSR file, the transaction continues to hold shared control of the control interval and causes transactions that attempt to update records in the same control interval to wait.

Similarly, transactions hold VSAM resources for too long if a EXEC CICS READ UPDATE or WRITE MASSINSERT is outstanding over a wait for terminal input.

Resource type FCRBWAIT—file recovery failed to complete

If your task is waiting on FCRBWAIT, it means that file recovery failed to complete. Refer the problem to the IBM Support Center.

Resource type FCTISUSP—wait for a VSAM transaction ID

If your task is waiting on resource type FCTISUSP, it means that there are no VSAM transaction IDs available. Transaction IDs are retained by a task for the duration of a MASSINSERT session.

Waits on FCTISUSP should not be prolonged, and if your task stays suspended on this resource type, it could indicate any of the following:

- There could be a system-wide problem. CICS could have stopped running, or it might be running slowly. Turn to Chapter 2, “Classifying the problem” on page 9 for advice if you suspect this.
- There could be a performance problem. Guidance about dealing with performance problems is given in Chapter 8, “Dealing with performance problems” on page 139.
- The logic of your applications might need changing, so that tasks do not retain VSAM transaction IDs for too long. If the task does other processing during the session, perhaps even involving input from an operator, code to release the VSAM transaction ID should be included each time.

Resource type FCXCWAIT—VSAM exclusive control wait

If your task is waiting on resource type FCXCWAIT, it means that it cannot get exclusive control of a VSAM control interval at the present time. Another task already has shared or exclusive control of the control interval, so your task is suspended pending the release of that control interval. An exclusive control wait on resource type FCXCWAIT occurs within CICS, unlike the similar wait on FCIOWAIT, which occurs within VSAM. See page 110.

If you find that exclusive control conflicts occur too often in your system, consider changing the programming logic so that applications are less likely to have exclusive control for long periods.

The possibility that a task is deadlocked, waiting on itself or another task for release of the control interval, is dealt with in the next section.

Exclusive control deadlock

Prior to CICS Transaction Server for VSE/ESA Release 1, a task could be made to wait on itself for exclusive control of a VSAM control interval. The task would be deadlocked, and could neither release exclusive control nor reacquire it.

Alternatively, a task could be made to wait on another task which has exclusive or shared control of a VSAM control interval. If this second task was, itself, waiting for exclusive control of a resource of which the first task has exclusive or shared control, then both tasks would be deadlocked.

In CICS Transaction Server for VSE/ESA Release 1, however, a mechanism exists to avoid exclusive control deadlock. If a task is waiting on resource type FCXCWAIT and causing the task to wait (causing a deadlock), the task is abended either with abend code AFCF or AFCG at the time that it makes the request for exclusive control.

A task that is abended with abend code AFCF would have been waiting for exclusive control of a VSAM control interval of which another task has shared or exclusive control.

A task that is abended with abend code AFCG would have been waiting for exclusive control of a VSAM control interval of which it has shared control.

See the *VSE/ESA Messages and Codes Volume 3* manual for further details of these abend codes.

To resolve the problem, determine which program caused the potential deadlock. Find out which programs are associated with the abended task, and attempt to find the one in error. It is likely to be one that provides successive browse and update facilities. When you have found the programs associated with the task, turn to “How tasks can become deadlocked waiting for exclusive control” on page 113 for guidance about finding how the error might have occurred.

How tasks can become deadlocked waiting for exclusive control

Tasks can become deadlocked waiting for exclusive control of a CI only when they have shared control of the CI and then attempt to get exclusive control without relinquishing shared control first. This can only occur for VSAM shared resource data sets.

For the deadlock to occur, a transaction must first issue a VSAM READ SEQUENTIAL request via EXEC CICS STARTBR. This is a VSAM "shared control" operation. It must then issue some VSAM request requiring exclusive control of the CI without first ending the shared control operation.

The requests that require exclusive control of the CI are:

- VSAM READ UPDATE, via EXEC CICS READ UPDATE and, subsequently, EXEC CICS REWRITE.

  Exclusive control of the CI is not acquired until after the initial read is complete, but it happens automatically after that and the CI is not released until the record has been rewritten.
- VSAM WRITE DIRECT, via EXEC CICS WRITE
- VSAM WRITE SEQUENTIAL, via EXEC CICS WRITE MASSINSERT

VSAM handles requests requiring exclusive control on a data set that is already being used in shared control mode by queueing them internally. VSAM returns control to CICS, but transactions waiting for exclusive control remain suspended.

Example of code causing an exclusive control deadlock

The following sequence of EXEC commands would cause an exclusive control deadlock to occur.

The first command causes shared control to be acquired:

```
EXEC CICS STARTBR
     FILE(myfile)
     RIDFLD(rid-area)
```

This causes no problems. The next command at first acquires shared control while the record is read into "input-area". When an attempt is subsequently made to get exclusive control, deadlock occurs because the task that wants exclusive control is also the task that is preventing it from being acquired.

```
EXEC CICS READ
     FILE(myfile)
     INTO(input-area)
     RIDFLD(rid-area)
     UPDATE
```

The following sequence of commands would not cause deadlock to occur, because the transaction relinquishes its shared control of the CI by ending the browse before attempting to get exclusive control of it.

The first command causes shared control to be acquired:

```
EXEC CICS STARTBR
     FILE(myfile)
     RIDFLD(rid-area)
```

The next command causes shared control to be relinquished:

```
EXEC CICS ENDBR
     FILE(myfile)
```

The next command initially causes shared control to be acquired. The record is read into “input-area”, and then exclusive control is acquired in place of shared control.

```
EXEC CICS READ
     FILE(myfile)
     INTO(input-area)
     RIDFLD(rid-area)
     UPDATE
```

The transaction now resumes. Exclusive control is relinquished following the next REWRITE or UNLOCK command on file “myfile”.

Resource type KC_ENQ—wait for a record lock in a recoverable VSAM file

When an application updates a record in a recoverable VSAM file, locking occurs at two levels. VSAM locks the CI when the record has been read, and CICS locks the record.

The CI lock is released as soon as the REWRITE (or UNLOCK) request is completed. However, the record is not unlocked by CICS until the updating transaction has reached a syncpoint. This is to ensure that data integrity is maintained if the transaction fails before the syncpoint and the record has to be subsequently backed out.

If a second transaction attempts to update the same record while it is still locked, it is suspended on resource type KC_ENQ until the lock is released. This can be a long wait, because the update might depend on a terminal operator typing in data. Also, the suspended transaction relinquishes its VSAM string and, perhaps, its exclusive control of the CI, and has to wait once more for those resources.

If transactions are commonly made to wait for this reason, you should review the programming logic of your applications to see if the record-locking time can be minimized.

Note that CICS only locks the recoverable record for update. Other transactions are allowed to read the record, and this presents a potential read integrity exposure. Thus, a transaction might read a record after an update has been made, but before the updating transaction has reached its syncpoint. If the reading transaction takes action based on the value of the record, the action will be incorrect if the record has to be subsequently backed out.

There is some more information about read integrity in Chapter 9, “Dealing with incorrect output” on page 149.

Resource type KC_ENQ—wait for exclusive control of a DAM record

DAM does not use the “control interval” concept. When a task reads a record for update, the record is locked so that concurrent changes cannot be made by two transactions. The lock is released at the end of the current logical unit of work. If a second task attempts to update the same record while the first has the lock, it is suspended on resource type KC_ENQ.

Interval control waits

Read this section if you have a task that is not running, and interval control seems to be involved. The following is a list of possible cases, and suggestions to consider before you carry out a detailed investigation. If these do not give you enough information to solve the problem, turn to “Finding the reason for a DELAY request not completing” on page 116 for further guidance.

If, in the course of your preliminary investigations, you find that the task is waiting because the terminal on which it is due to start is unavailable, turn to “Terminal waits” on page 94.

- A terminal task that should have been initiated with an EXEC CICS START command did not start when you expected it to. CEMT INQUIRE TASK does not recognize the task, because it has not yet been attached.

  Identify the terminal where the subject task should have started, and see if that terminal is, for some reason, unavailable. You can use CEMT INQUIRE TERMINAL to find the status of the terminal.

- A task is waiting on resource type ICGTWAIT. This means that the task has issued an EXEC CICS RETRIEVE WAIT command, and the data to be retrieved is not available. The resource name gives you the name of the terminal running the task in the ICGTWAIT wait and therefore the target TERMID for other tasks issuing EXEC CICS START commands to supply more data. If there are no tasks in the system that would issue START commands for this TERMID, you need to determine whether this is reasonable. If there are such tasks in the system, check to see why they are not issuing the required START commands. They might, for example, be waiting for terminal input.

  Look, too, at the deadlock time-out interval (DTIMOUT) and the system purge value (SPURGE) for the task issuing the EXEC CICS RETRIEVE WAIT command. If there is no DTIMOUT value or SPURGE(NO) has been specified on the RDO TRANSACTION resource definition, the task will wait indefinitely for the data.

  Note: The task waiting on resource ICGTWAIT might not be the one that you first set out to investigate. Any AID task scheduled to start at the same terminal cannot do so until the current task has terminated.

- You have found that the task is waiting on resource type ICWAIT. This means that the task issued an EXEC CICS DELAY command that has not yet completed. Check that the interval or time specified on the request was what you intended. If you believe that the expiry time of the request has passed, then this suggests a possible CICS error.

  Consider, too, the possibility that the task was the subject of a long DELAY that was due to be canceled by some other task. If the second task failed before it

could cancel the delay, the first would not continue until the full interval specified on DELAY had expired.

- A task that issued EXEC CICS POST did not have its ECB posted when you expected it to.

  Check to make sure the interval or time you specified was what you intended.

- A task that issued EXEC CICS WAIT EVENT was not resumed when you thought it should have been.

  Assuming the WAIT was issued sometime after a POST, first check to make sure that the interval or time specified on the POST was what you intended. If it was, next check to see whether the ECB being waited on was posted. If it was, that indicates a possible CICS error.

If none of the simple checks outlined here help you to solve the problem, read the next section.

Finding the reason for a DELAY request not completing

If your preliminary investigations have not shown the reason for the wait, you need to look in greater detail at the evidence available. Take a system dump, and format it using the keywords CSA, ICP, and AP. These provide the common system area, the interval control program control blocks, and the task control areas, respectively. You might also find information given by the formatting keywords KE (kernel storage areas, including the calling sequence for each task), DS (dispatcher task summary, including details of suspended tasks), and TR (internal trace table) to be useful.

First, locate field CSATODTU in the CSA. Make a note of the value there, which is the current CICS time of day in internal 'timer units'. Now locate the TCA for your task, and read the value of field TCAICEAD. This gives you the address of the interval control element for your task. Use this information to find the ICE (interval control element) for the task, and look at field ICEXTOD. Make a note of the value there.

If ICEXTOD is greater than CSATODTU, the ICE has not yet reached the expiry time. These are the possible explanations:

- Your task either did not make the DELAY request you expected, or the interval specified was longer than intended. This could indicate a user error. Check the code of the transaction issuing the request to make sure it is correct.
- Your task's delay request was not executed correctly. This might indicate an error within CICS code, or a corrupted control block.

If ICEXTOD is equal to CSATODTU (very unlikely), you probably took the system dump just as the interval was about to expire. In such a case, attempt to re-create the problem, take another system dump, and compare the values again.

If ICEXTOD is less than CSATODTU, the ICE has already expired. The associated task should have resumed. This indicates that some area of storage might have been corrupted, or there is an error within CICS code.

Using trace to find out why tasks are waiting on interval control

Before using trace to find out why your task is waiting on interval control, you need to select an appropriate trace destination and set up the right tracing options.

By their nature, interval control waits can be long, so select auxiliary trace as the destination, because you can specify large trace data sets for auxiliary trace. However, the data sets do not have to be large enough to record tracing for the whole interval specified when you first detected the problem. That is because the error is likely to be reproducible when you specify a shorter interval, if it is reproducible at all. For example, if the error was detected when an interval of 20 seconds was specified, try to reproduce it specifying an interval of 1 second.

As far as tracing selectivity is concerned, you need to capture level-2 trace entries made by dispatcher domain, timer domain, and interval control program. Use the CETR transaction to set up the following tracing options:

1. Specify special tracing for the level-2 trace points for components DS (dispatcher domain), TI (timer domain), and IC (interval control program).
2. Select special tracing for the task causing the problem, by specifying special tracing both for the transaction and for the terminal where it is to be run.
3. Set the master system trace flag off, to turn off all standard tracing. This helps minimize the number of trace entries not connected with the problem.
4. Make sure that auxiliary tracing is active, then set the transaction running. When the problem appears, format the auxiliary trace data set and either print it or view it online.

The sort of trace entries that you can expect in normal operation are shown in the figures that follow. They show the flow of data and control following execution of the command EXEC CICS DELAY INTERVAL(000003). A similar set of trace entries would be obtained if TIME had been specified instead of INTERVAL, because TIME values are converted to corresponding INTERVAL values before timer domain is called.

Figure 10 shows the first two entries that you get following execution of the EXEC CICS DELAY INTERVAL(000003) command.

```
AP 00E1 EIP ENTRY DELAY                              REQ(0004) FIELD-A(0034BD70 ....) FIELD-B(08001004 ....)
             TASK-00163 KE_NUM-0007 TCB-009F3338 RET-8413F43E TIME-16:31:58.0431533750 INTERVAL-00.0000166250    =000602=
AP 00F3 ICP ENTRY WAIT                               REQ(2003) FIELD-A(0000003C ....) FIELD-B(00000000 ....)
             TASK-00163 KE_NUM-0007 TCB-009F3338 RET-84760B88 TIME-16:31:58.0432681250 INTERVAL-00.0000370000    =000605=
```

Figure 10. Trace entries following EXEC CICS DELAY INTERVAL(000003) invocation

Notes:

1. Trace point **AP 00E1** is on ENTRY to the EIP DELAY routine. The function is stated in the trace header, and the fact that this trace is made on ENTRY can be deduced from the value shown in the request field, REQ(0004).

   The rightmost two bytes of FIELD B give the EIBFN value, in this case X'1004'. This shows that this is an interval control DELAY request.

   The value shown against TASK is the trace task number, and it is unique to the task while the task is in the system. Its purpose is to show which trace entries relate to which tasks. The task number in this example is 00163. As long as the task is in the system, and either running or suspended, trace entries having

this task number always relate to it. Use the task number for your task to identify the trace entries associated with it.

2. Trace point **AP 00F3** is on ENTRY to the ICP WAIT routine. The function is given explicitly in the trace header, and both the function and the fact that this represents ENTRY to the routine can be deduced from the request field, REQ(2003).

   The value of FIELD A, X'0000003C', is an important one for problem determination. It shows the interval that has been specified, in this case three seconds. Check the value shown here for your own task, to make sure it is what you expect it to be.

Look next for an entry with point ID DS 0004 showing your task being suspended, as in Figure 11. You might see TI domain trace entries preceding it that show entry and exit for FUNCTION(REQUEST_NOTIFY_INTERVAL), but these do not always appear.

There might also be some intervening entries, but they are unlikely to be of relevance to the problem.

```
TI 0100 TISR  ENTRY - FUNCTION(REQUEST_NOTIFY_INTERVAL) DOMAIN_TOKEN(00E70000 , 00000000) STCK_INTERVAL(00000002DC6C1000)
                  PERIODIC_NOTIFY(NO) NOTIFY_TYPE(TIMER_TASK)
              TASK-00163 KE_NUM-0007 TCB-009F3338 RET-8476352A TIME-16:31:58.0442390000 INTERVAL-00.0000155000    =000614=
                1-0000  00600000 00000006 00000000 00000000  B3B00000 00000000 01000000 00000000  *.-.............................*
                  0020  00000000 00000000 00000000 00E70000  00000000 00000000 00000000 00000002  *.............X.................*
                  0040  DC6C1000 00000000 02020000 00000000  00000000 00000000 00000000 00000000  *.%.............................*
TI 0101 TISR  EXIT  - FUNCTION(REQUEST_NOTIFY_INTERVAL) RESPONSE(OK) TIMER_TOKEN(03B9B058 , 0000001B)
              TASK-00163 KE_NUM-0007 TCB-009F3338 RET-8476352A TIME-16:31:58.0738898750 INTERVAL-00.0296188125*   =000617=
                1-0000  00600000 00000006 00000000 00000000  B3B00000 00000000 01000100 00000000  *.-.............................*
                  0020  00000000 00000000 00000000 00E70000  00000000 03B9B058 0000001B 00000002  *.............X.................*
                  0040  DC6C1000 00000000 02020000 00000000  00000000 00000000 00000000 00000000  *.%.............................*
DS 0004 DSSR  ENTRY - FUNCTION(SUSPEND) SUSPEND_TOKEN(01040034) RESOURCE_NAME(1477) RESOURCE_TYPE(ICWAIT) PURGEABLE(YES)
                  DEADLOCK_ACTION(INHIBIT)
              TASK-00163 KE_NUM-0007 TCB-009F3338 RET-847645CE TIME-16:31:58.0739336250 INTERVAL-00.0000437500    =000618=
                1-0000  00580000 00000014 00000001 00000000  B7050000 00000000 04000100 00000000  *.............................*
                  0020  00000000 01040034 F1F4F7F7 40404040  C9C3E6C1 C9E34040 0000001B 00000002  *........1477    ICWAIT  ........*
                  0040  DC6C1000 00000000 02010003 00000000  00000000 00000000                   *.%.....................        *
```

Figure 11. Trace entries showing interval calculation and task suspension

Notes:

1. Trace point **TI 0100**, if shown, is on ENTRY to the REQUEST_NOTIFY_INTERVAL function of timer domain. This is stated explicitly in the trace header.

   The value shown in the header for STCK_INTERVAL is derived from the machine store clock value calculated for the DELAY interval specified on the EXEC CICS DELAY command. You can find out how store clock values are related to times in hours, minutes, and seconds from the *ESA/390 Principles of Operation* manual.

   If you do the calculation, you will find that the value shown is not exactly equal to the interval you specified. An extra microsecond is added, to account for the case where the interval is specified as 0.

   In this example, 3 seconds is exactly equal to a store clock interval of X'00000002DC6C0000'. You can see that the actual store clock value is quoted in the trace entry as X'00000002DC6C1000', which is 3 seconds *plus* 1 microsecond.

The TIME field of the trace entry shows the time at which the entry was made, in the format hh:mm:ss. The value in this example (ignoring the fractions of a second) is 16:31:58. It follows that the task is due to be resumed when the time is 16:32:01, because the interval is 3 seconds.

2. Trace point **TI 0101**, if shown, is on EXIT from the REQUEST_NOTIFY_INTERVAL function of timer domain.

   You can see from RESPONSE(OK) in the header that the function completed normally.

3. Trace point **DS 0004** is on ENTRY to the dispatcher task SUSPEND/RESUME interface.

   The SUSPEND_TOKEN field in the trace header is significant. It shows the unique suspend token being used for this SUSPEND/RESUME dialog, and it is referred to explicitly again in a later trace entry showing that the task has been resumed. In this example, the suspend token is X'01040034'.

   Any subsequent dispatcher trace entry that shows the suspend token for your task is connected with the suspension or resumption of the task.

   Field RESOURCE_TYPE(ICWAIT) in the trace header shows that the resource type associated with this suspend is ICWAIT. ICWAIT is the resource type that is returned on CEMT INQUIRE TASK for tasks that are waiting on interval control.

Next, obtain some trace entries that record system activity during the period when your task is suspended. There are likely to be relatively few at the level of tracing detail you have specified, but you need to look further on in the trace to find the next entries of interest.

Add 3 seconds (or whatever interval you specified) to the time shown on the last trace entry you looked at, and turn forward to the trace entries made at around that time. Now look for an entry made from trace point DS 0004. This does not show the task number for your task, but it does show its suspend token. When you have found it, go back one entry. You should find there a trace entry made from trace point AP F322. This and the following two trace entries of interest are shown in Figure 12.

```
AP F322 APTIX RESUMED - SYSTEM TASK APTIX RESUMED
              TASK-00006 KE_NUM-0009 TCB-009F3338 RET-84773724 TIME-16:32:01.1016870625 INTERVAL-00.0001065000    =000670=
                1-0000  01000000 D7C5D5C4 D5D6E3D7 01107739  00E70000 00000000 03B9B058 0000001B  *....PENDNOTP.....X.............*
                  0020  01080002 00D40000 00000000 03B9B000  00000001 01050002 00000000 00000000  *.....M.........................*
                  0040  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
                  0060  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
DS 0004 DSSR  ENTRY - FUNCTION(RESUME) SUSPEND_TOKEN(01040034)
              TASK-00006 KE_NUM-0009 TCB-009F3338 RET-847646D4 TIME-16:32:01.1019761875 INTERVAL-00.0000278125    =000674=
                1-0000  00580000 00000014 00000001 00000000  B4000000 00000000 05000100 00000000  *................................*
                  0020  00000000 01040034 00000000 00E70000  00000000 03B9B058 0000001A 000026EE  *.............X..................*
                  0040  D9AC1000 00000000 00000000 0001632C  00000000 00000000                    *R.....................        *
DS 0005 DSSR  EXIT - FUNCTION(RESUME) RESPONSE(OK)
              TASK-00006 KE_NUM-0009 TCB-009F3338 RET-847646D4 TIME-16:32:01.1019959375 INTERVAL-00.0000197500    =000675=
                1-0000  00580000 00000014 00000001 00000000  B4000000 00000000 05000100 00000000  *................................*
                  0020  00000000 01040034 00000000 00E70000  00000000 03B9B058 0000001A 000026EE  *.............X..................*
                  0040  D9AC1000 00000000 00000000 0001632C  00000000 00000000                    *R.....................        *
```

Figure 12. Trace entries showing your task being resumed

Notes:

1. Trace point **AP F322** is used to report that system task APTIX has been resumed. APTIX has the job of “waking up” your task on expiration of the specified interval.

   The task number for APTIX is, in this case, X'00006', and this value is shown on the trace entry.

2. Trace point **DS 0004** is on entry to the dispatcher SUSPEND/RESUME interface. This function is stated explicitly in the header.

   TASK-00006 indicates that the trace entry is for system task APTIX.

   SUSPEND_TOKEN(01040034) shows that APTIX is requesting dispatcher domain to resume the task that was suspended for the specified interval. You will recall that a suspend token of X'01040034' was given to your task when it was first suspended.

3. Trace point **DS 0005** is on exit from the dispatcher SUSPEND/RESUME interface.

   The trace entry shows RESPONSE(OK), indicating that the task whose suspend token was X'01040034' has successfully been resumed. However, note that this does not necessarily mean that the task has started to run—it has only been made “dispatchable”. For example, it still needs to wait for a TCB to become available.

Now look forward in the trace, and locate a trace entry made from trace point AP 00F3 and showing your task number. This and the next entry conclude the DELAY request for your task. They are shown in Figure 13.

```
AP 00F3 ICP EXIT NORMAL                              REQ(0005) FIELD-A(01000300 ....) FIELD-B(03BD6EE0 ..>.)
             TASK-00163 KE_NUM-0007 TCB-009F3338 RET-84760B88 TIME-16:32:01.1023045625 INTERVAL-00.0000154375    =000688=
AP 00E1 EIP EXIT DELAY OK                            REQ(00F4) FIELD-A(00000000 ....) FIELD-B(00001004 ....)
             TASK-00163 KE_NUM-0007 TCB-009F3338 RET-8413F43E TIME-16:32:01.1024153750 INTERVAL-00.0000235625    =000691=
```

Figure 13. Trace entries showing satisfactory conclusion of the DELAY request

Notes:

1. Trace point **AP 00F3** is on EXIT from interval control program. Field REQ(0005) shows that this is so, and it also shows that the response was normal. Anything other than a normal response would result in a value other than X'00' for the first byte of the REQ field.

2. Trace point **AP 00E1** is on EXIT from the EXEC interface program. This is shown by bits 0–3 of the second byte of the REQ value, X'F4'.

   The values shown for FIELD A and FIELD B show that no exception condition was detected.

That is the end of the DELAY processing, and the task that was suspended should have been resumed.

When you look at your own trace table, concentrate on finding the point at which the processing went wrong. Also, watch for bad parameters. If you do find one, it could mean that an application has a coding error, or some field holding a parameter has been overlaid, or an error has occurred in CICS code.

Checking your application code is the easiest option you have. If you find that it is correct and you suspect a storage violation, see Chapter 10, “Dealing with storage violations” on page 173. If you think the error is in CICS code, contact the IBM Support Center.

XRF alternate system waits

The XRF takeover process involves several operations. Before each operation can be started, one or more events must have completed. For example:

- Passively-shared data sets must not be opened until it is known that the active system has terminated.

  Note: This is the only way an alternate system can be sure that no more data will be written by an active system.

- Resource managers, such as transient data and temporary storage, rely on the time-of-day clock providing them with a nondecreasing value to ensure the proper management of their resources. The alternate system must not restart a resource manager until the alternate time-of-day clock has been synchronized with the active time-of-day clock.

A system task that issued a takeover request to CAVM waits on the ECB WCSTCECB in the CAVM static control block (DFHWCGPS) until CAVM has decided to accept or reject the request. The DFHDSSR WAIT_EXTERNAL request is issued in DFHWSRTR. The CAVM subtask posts WCSTCECB in either DFHWSTKV (the normal case) or DFHWSSOF (CAVM failure).

The following ECBs each represent an event. The ECBs are located in the static storage for DFHXRP. The ECBs and the events are:

- XRSTIECB—the CAVM has initiated a takeover.
- XRSIAECB—the alternate system is now the incipient active system.
- XRSTCECB—the active system is known to have terminated.
- XRSRAECB—DFHRSD is available for use by DFHRC (recovery control) macros.
- XRSSSECB—the time-of-day clock is synchronized with active system sign off.
- XRSSTECB—the time-of-day clock is synchronized with active system termination.

XRSTIECB

This ECB is posted by DFHXRA, following a successful call to the CAVM to initiate takeover. Once the ECB has been posted, DFHXRA attaches a system transaction to initiate the switch of terminals with backup sessions. DFHXRA is called from either the surveillance task (DFHXRSP), or the console communication task (DFHXRCP). No tasks wait for XRSTIECB to be posted.

XRSIAECB

The XRSIAEB ECB is posted by DFHXRA, following notification by the CAVM that an alternate system is now the incipient active system. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSIAECB to be posted.

XRSTCECB

The XRSTCECB ECB is posted by DFHXRA, following notification by the CAVM that an active system has terminated. There can be a delay in posting the ECB if:

- An SDUMP is being taken as part of the active system termination process.
- In a 2-CPC environment, the active CPC has failed, and the operator failed to reply to the messages sent to the console.

No tasks wait for XRSTCECB to be posted.

XRSRAECB

The XRSRAECB ECB is posted by DFHXRA once the restart data set has been opened for DFHRC use. No tasks wait for XRSRAECB to be posted.

XRSSSECB

The XRSSSECB ECB is posted by DFHXRA following notification by the CAVM that the time-of-day clock is synchronized with active sign off. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSSSECB to be posted.

XRSSTECB

The XRSSTECB ECB is posted by DFHXRA, following notification by the CAVM that the time-of-day clock is synchronized with respect to active system termination. There may be a delay in posting the ECB if the time indicated by the active system time-of-day clock is significantly ahead of that indicated by the alternate system time-of-day clock. DFHXRA is called from the surveillance task (DFHXRSP).

Only the system initialization task, DFHSII1, waits for XRSSTECB to be posted. When the ECB is posted, DFHSII1 links to DFHJRCP to restore journal states before attaching the remaining resource manager tasks.

XRPUTMSG

You are only likely to find either of the CICS-supplied transactions CEDA or CESN waiting on a resource type of XRPUTMSG, and only during XRF takeover by the alternate CICS system. It can indicate either of these conditions:

- Data that is required by the transactions is held on a data set subject to a VSE LOCK, and another job currently has the lock.
- There is an error in the CICS availability manager.

If it seems clear that a VSE LOCK is not implicated, refer the problem to the IBM Support Center.

CICS system task waits

From an analysis of trace, you could have evidence that a CICS system task is in a wait state. You might have seen the task suspended on a SUSPEND call to the dispatcher, but with no corresponding RESUME call. Alternatively, by looking at the dispatcher task summary in a formatted CICS system dump, you might see that a CICS system task is waiting.

Note: You cannot get online information about waiting system tasks from CEMT INQUIRE TASK or EXEC CICS INQUIRE TASK.

If a system task is in a wait state, and there is a system error preventing it from resuming, contact your IBM Support Center. However, do not assume that there is a system error unless you have other evidence that the system is malfunctioning. Other possibilities are:

- Some system tasks are intended to wait for long periods while they wait for work to do. Module DFHSMSY of storage manager domain, for example, can stay suspended for minutes, or even hours, in normal operation. Its purpose is to clean up storage when significant changes occur in the amount being used, and that might happen only infrequently in a production system running well within its planned capacity.
- System tasks perform many I/O operations, and they are subject to constraints like string availability and volume and data set locking. In the case of tape volumes, the tasks can also be dependent on operator action while new volumes are mounted.

If, in addition to the waiting system task, you think you have enough evidence to show that there is a system error, contact your IBM Support Center.

FEPI waits

This section outlines the CICS waits that FEPI issues. Table 23 shows the points at which FEPI issues CICS waits:

Table 23. FEPI waits

Resource name	Resource type	Wait type	Description
FEPI_RQE	ADAPTER	WAIT_EXTERNAL	Issued in the FEPI adapter when a FEPI command is passed to the Resource Manager for processing. Ends when the Resource Manager has processed the request.
SZRDP	FEPRM	WAIT_EXTERNAL	Issued in the FEPI Resource Manager when it has no work to do. Ends when work arrives (from either the FEPI adapter or a VTAM exit).

It is possible for a FEPI_RQE wait to be outstanding for a long time—such as when awaiting a flow from the back-end system that is delayed due to network traffic. It is recommended that you do *not* cancel tasks that are waiting at this point; to do so could lead to severe application problems.

An SZRDP wait is generated when the FEPI Resource Manager is idle. Consequently, the SZ subtask is also inactive. On lightly loaded systems, this occurs frequently.

If the Resource Manager abends, then any active CICS FEPI transactions are left waiting on the FEPI_RQE resource. Because the Resource Manager is absent, these waits never get posted, so the transactions suspend. You must issue a CEMT SET TASK FORCEPURGE command to remove these suspended transactions from the system.

What to do if CICS has stalled

CICS can stall during initialization, when it is running apparently “normally”, or during termination.

These possibilities are dealt with in:

- “CICS has stalled during initialization”
- “CICS has stalled during a run” on page 125
- “CICS has stalled during termination” on page 127

If XRF takeover by an alternate CICS system fails to complete satisfactorily, that might also appear to you as a CICS stall. See Chapter 11, “Dealing with XRF errors” on page 185 for more information.

CICS has stalled during initialization

If CICS stalls during initialization, on cold start, warm start, or emergency restart, the first place to look is the VSE console log. This tells you how far initialization has progressed.

Note that there might be significant delays at specific stages of initialization, depending on how CICS last terminated.

On cold start, loading the GRPLIST definitions from the CSD data set can take several minutes. For large systems, this delay could be 20 minutes or more. You can tell if this stage of initialization has been reached because you get the following console message for each list specified on the GRPLIST system initialization parameter:

```
DFHSI1511 INSTALLING GROUP LIST xxxxxxxx
```

On warm start, there may be a considerable delay while resource definitions are being created from the global catalog. You are not told explicitly that this processing is taking place, but it occurs just after this message is issued:

```
DFHJC4500 - nn OF nn JOURNALS SUCCESSFULLY OPENED
```

If you find that unexpected delays occur at other times during CICS initialization, consider the messages that have already been sent to the console and see if they suggest the reason for the wait. For example, a shortage of storage is one of the most common causes of stalling, and is always accompanied by a message. The JCL job log is another useful source of information.

You can find out if this has happened by taking an SDUMP of the CICS region. Format the dump using the keywords KE and DS, to get the kernel and dispatcher task summaries.

Consider, too, whether any first- or second-stage program list table (PLT) program that you have written could be in error. If such a program does not follow the strict protocols that are required, it can cause CICS to stall. For programming information about PLT programs, see the *CICS Customization Guide*.

CICS has stalled during a run

If a CICS region that has been running normally stalls, so that it produces no output and accepts no input, the scope of the problem is potentially system-wide. The problem might be confined exclusively to CICS, or it could be caused by any other task running under VSE.

Look first on your VSE console for any messages. Look particularly for messages indicating that operator intervention is needed, for example to change a tape volume. The action could be required on behalf of a CICS task, or it could be for any other program that CICS interfaces with.

If there is no operator action outstanding, inquire on active users at the VSE console to see what the CPU usage is for CICS. If you find the value is very high, this probably indicates that a task is looping. Read Chapter 7, “Dealing with loops” on page 129 for advice about investigating the problem further.

If the CPU usage is low, CICS is doing very little work. Some of the possible reasons are:

- The system definition parameters are not suitable for your system.
- The system is short on storage, and new tasks cannot be started. This situation is unlikely to last for long unless old tasks cannot, for some reason, be purged.
- The system is at one of the MXT or transaction class limits, and no new tasks can be attached. In such a case, it is likely that existing tasks are deadlocked, and for some reason they cannot be timed out.
- There is an exclusive control conflict for a volume.
- There is a problem with the communications access method.
- There is a CICS system error.

To find out if any of these apply to your system, see the paragraphs that follow. For some of the investigations, you need to see a system dump of the CICS partition. If you do not already have one, you can request one using the VSE console. Make sure that CICS is apparently stalled at the time you take the dump, because otherwise it will not provide the evidence you need. Format the dump using the formatting keywords KE and XM, to get the storage areas for the kernel and the transaction manager.

Are the system definition parameters wrong?

It is possible that the system definition parameters for your system are causing it to stall, possibly at a critical loading. Take a look at what has been specified, paying particular attention to these items:

- The CICS maximum tasks (MXT) and transaction class (MAXACTIVE) limits. If these are too low, new tasks could fail to be attached. If you suspect one of these limits is the cause of the stall, read “Are MXT or transaction class limits causing the stall?” on page 126 for a way of getting further evidence.
- ICV, the system region exit time. If this is set too high, CICS might relinquish control to the operating system for longer than intended when it has no work to do, perhaps giving the impression of a stall.

- ICVR, the runaway task time interval. If this is set too high, a runaway task could stop other tasks from running for a relatively long time. It can have a value up to 2 700 000 milliseconds, in which case a runaway task would not time out for 45 minutes. CICS could, in the meantime, be stalled. If the ICVR is set to 0, the runaway task does not time out at all.

  You should already have an indication if the ICVR is the problem, from the CPU usage (see above).
- ICVTSD, the terminal scan delay interval. The effect differs for VTAM and non-VTAM terminals, but if its value is not appropriate for your system the effect could make you think that CICS is stalled.

For more details about the choice of these and other system definition parameters, see the *CICS Performance Guide*.

Is the system short on storage?

If storage is under stress, storage manager statistics indicate that a storage stress situation has occurred ('Times went short on storage'). In addition, if the SOS is caused by a suspended GETMAIN or if CICS is unable to alleviate the situation by releasing programs with no current user, and slowing the attachment of new tasks:

- A message is sent to the console saying that CICS is short on storage (DFHSM0131I for storage below the 16 megabyte line, DFHSM0133I for storage above the 16 megabyte line)
- The storage manager statistic 'Times went short on storage' is updated.

CICS can go short on storage independently in any DSA. You may see tasks suspended on any of the resource types, CDSA, SDSA, RDSA, UDSA, ECDSA, ESDSA, ERDSA, or EUDSA.

Are MXT or transaction class limits causing the stall?

Before new transactions can be attached for the first time, they must qualify under the MXT and transaction class limits. In a system that is running normally, tasks run and terminate and new transactions are attached, even though these limits are reached occasionally. It is only when tasks can neither complete nor be purged from the system that CICS can stall as a result of one of these limits being reached.

Look first at the transaction manager summary in the formatted system dump.

Investigate the tasks accepted into the MXT set of tasks to see if they are causing the problem. XM dump formatting formats the state of MXT and provides a summary of the TCLASSes and of the transactions waiting for acceptance into each TCLASS.

Now look at the task control queue control area, in the AP section of the dump for a summary of task enqueues and resources. If you find that many tasks are enqueued on the same resource, this could indicate a deadlock involving tasks that are not purgeable. Investigate these tasks to find if they are causing the problem.

Is there a problem with the communications access method?

If you suspect that there is a communication problem, you can inquire on the status of VTAM from the VSE console. To do this, use the command `MSG nn,DATA=CEMT INQUIRE VTAM`. You can use this command only if the VSE console has been defined to CICS as a terminal. The status returned has a value of OPEN or CLOSED.

- If the VTAM status is OPEN, the problem could be associated with processing done in the VTAM part of your system or with processing done in the CICS part of your system. If it appears that there is a communication problem, consider using VTAM buffer tracing. For guidance about using this technique, see Chapter 14, “Using traces in problem determination” on page 199.
- If the VTAM status is CLOSED, CICS cannot use VTAM to perform communication functions.

Is there a CICS system error?

If you have investigated all the task activity, and all the other possibilities from the list, and you have still not found an explanation for the stall, it is possible that there is a CICS system error. Contact the IBM Support Center with the problem.

CICS has stalled during termination

Waits often occur when CICS is being quiesced because some terminal input or output has not been completed. To test this possibility, try using the CEMT transaction to inquire on the tasks currently in the system.

CICS termination takes place in two stages:

1. All transactions are quiesced.
2. All data sets and terminals are closed.

If you find that you cannot use the CEMT transaction, it is likely that the system is already in the second stage of termination. CEMT cannot be used beyond the first stage of termination.

Note: Even if CEMT is not included in the transaction list table (XLT), you can still use it in the first stage of termination.

The action to take next depends on whether you can use the CEMT transaction, and if so, whether or not there are current user tasks.

- **If you can use the CEMT transaction:**
  - If there are user tasks currently in the system, check what they are. A task may be performing a prolonged termination routine, or it might be waiting on a resource before it can complete its processing. It is also possible that a task is waiting for operator intervention.

    Determine what type of terminal is associated with the task. If the terminal is a 3270 device, some keyboard input might be expected. If it is a printer, it might have been powered off or it might have run out of paper.
  - If there are no user tasks in the system, it may be that one or more terminals have not been closed. Use the CEMT transaction to see which terminals are currently INSERVICE, and then use CEMT SET to place them OUTSERVICE.

If these actions fail, proceed as if you were unable to use the CEMT transaction.

- **If you cannot use the CEMT transaction**, go to the VSE console or the NetView® master terminal and display the active sessions. If necessary, close down the network using the `VARY NET,INACT,ID=applid` command. This should enable CICS to resume its termination sequence. If it does not, you might need to cancel the CICS job. If this does happen, consider whether any PLT program running in the second quiesce stage could be in error. If such a program did not follow the strict protocols that are required, it could cause CICS to stall during termination. For programming information about PLT programs, see the *CICS Customization Guide*.

Chapter 7. Dealing with loops

A loop is a sequence of instructions that is executed repetitively. Loops that are coded into applications must always be guaranteed to terminate, because otherwise you could get any of the symptoms of loops described in Chapter 2, “Classifying the problem” on page 9.

If a loop does not terminate, it could be that the termination condition can never occur, or it might not be tested for, or the conditional branch could erroneously cause the loop to be executed over again when the condition is met.

This chapter outlines procedures for finding which programs are involved in a loop that does not terminate, and contains the following topics:

- “What sort of loop is indicated by the symptoms?”
- “Investigating loops that cause transactions to abend with abend code AICA” on page 134
- “Investigating loops that are not detected by CICS” on page 136
- “What to do if you cannot find the reason for a loop” on page 138

If you find that the looping code is in one of your applications, check through the code to find out which instructions are in error. If it looks as if the error is in CICS code, you probably need to contact the IBM Support Center.

Some CICS domains can detect loops in their own routines, and let you know if one is suspected by sending the following message:

DFHxx0004 *applid* **A possible loop has been detected at offset X'** *offset* **' in module** *modname*.

The two characters “xx” represent the two-character domain index. If, for example, monitoring domain had detected the loop, the message number would be DFHMN0004.

If you see this sort of message repeatedly, contact the IBM Support Center.

What sort of loop is indicated by the symptoms?

Unplanned loops can be divided into those that can be detected by CICS, and those that cannot. Figure 14 shows a hierarchical classification of loops.

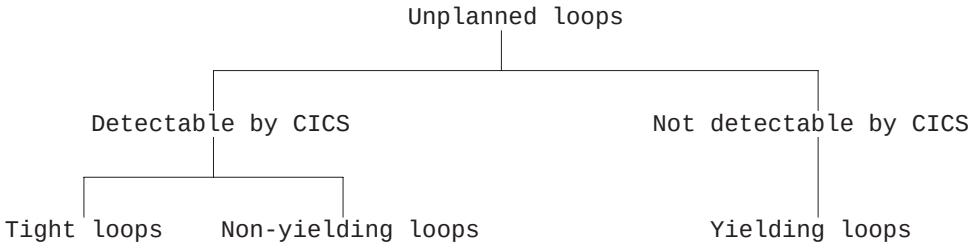


Figure 14. Hierarchical classification of loops

Figure 15 gives a specific example of code containing a simple tight loop.

© Copyright IBM Corp. 1979, 1999

```
PROCEDURE DIVISION.
    EXEC CICS
        HANDLE CONDITION ERROR(ERROR-EXIT)
               ENDFILE(END-MSG)
    END-EXEC.
ROUTE-FILE.
    EXEC CICS
        ROUTE INTERVAL(0)
              LIST(TERM-ID)
    END-EXEC.
 NEW-LINE-ATTRIBUTE.
    GO TO NEW-LINE-ATTRIBUTE.
    MOVE LOW-VALUES TO PRNTAREA.
    MOVE DFHBMPNL TO PRNTAREA.
```

Figure 15. Example of code containing a tight loop

CICS can detect some looping tasks by comparing the length of time the tasks have been running with the runaway time interval, ICVR, that you code in the system initialization table. If a task runs for longer than the interval you specify, CICS regards it as “runaway” and causes it to abend with an abend code of AICA.

However, in some cases, CICS requests that are contained in the looping code can cause the timer to be reset. Not every CICS request can do this; it can only happen if the request can cause the task to be suspended. Thus, if the looping code contains such a request, CICS cannot detect that it is looping.

The properties of the different types of loop, and the ways you can investigate them, are described in the sections that follow.

Tight loops and non-yielding loops

Tight loops and non-yielding loops are both characterized by the fact that the looping task can never be suspended within the limits of the loop. This makes them detectable by CICS, which compares the time they have been running continually with the runaway time interval, ICVR, that you code in the system initialization table. If the tasks run for longer than the interval you specify, CICS regards them as “runaway” and causes them to abend with an abend code of AICA.

Note: If you make the ICVR value equal to 0, runaway task detection is disabled. Runaway tasks can then cause the CICS partition to stall, meaning that CICS must be canceled and brought up again. You might choose to set ICVR to zero in test systems, because of the wide variation in response times. However, it is usually more advisable to set ICVR to a large value in test systems.

A tight loop is one involving a single program, where the same instructions are executed repeatedly and control is never returned to CICS.

Figure 16 on page 131 shows a tight loop, with an indefinite number of instructions contained in the loop. None of the instructions returns control to CICS. In the extreme case, there could be a single instruction in the loop, causing a branch to itself.

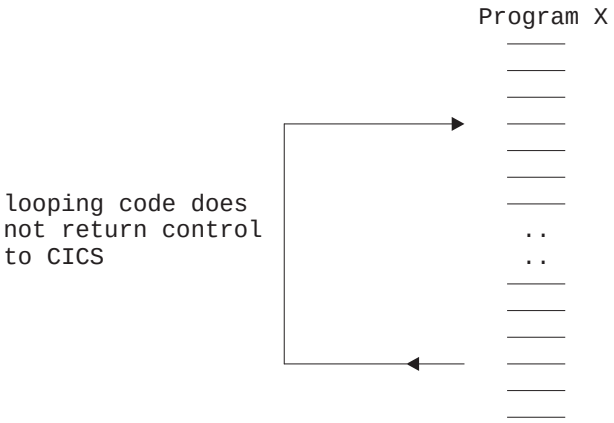


Figure 16. A tight loop

A non-yielding loop is also contained in a single program, but it differs from a tight loop in that control is returned temporarily from the program to CICS. However, the CICS routines that are invoked are ones that can neither suspend the program nor pass control to the dispatcher.[1] There is, therefore, no point at which the task can be suspended, and so the ICVR cannot be reset.

Figure 17 shows a non-yielding loop.

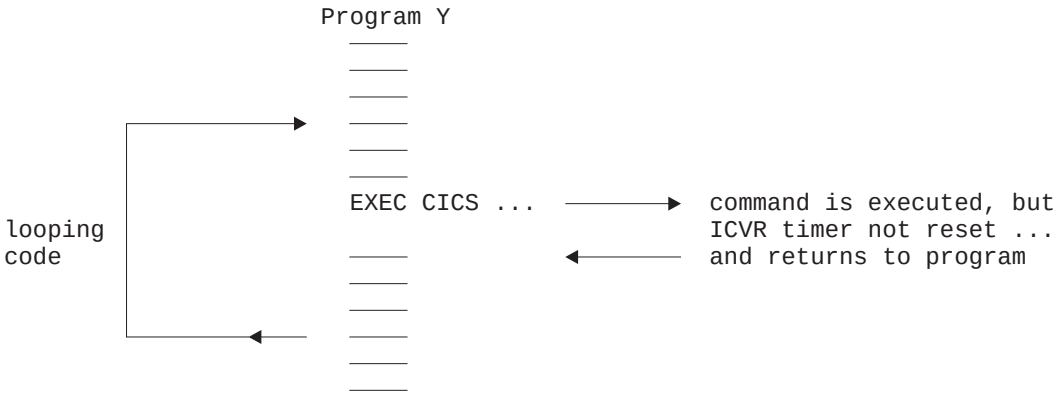


Figure 17. A non-yielding loop

[1] The CICS commands that do not cause tasks to wait include (but are not restricted to) ASKTIME, DEQ, ENQ, ENTER TRACENUM, FREEMAIN, HANDLE, RELEASE, TRACE ON/OFF. Whether a command allows the ICVR to be reset might also depend on other factors. For instance, a FREEMAIN might reset the ICVR if the storage lock is held. A READ might also not wait if the desired record is already in a VSAM buffer.

Figure 18 shows an example of code that contains a simple non-yielding loop. In this case, the loop contains only one CICS command, EXEC CICS ASKTIME.

```
PROCEDURE DIVISION.
    EXEC CICS
        HANDLE CONDITION ERROR(ERROR-EXIT)
              ENDFILE(END-MSG)
    END-EXEC.
ROUTE-FILE.
    EXEC CICS
        ROUTE INTERVAL(0)
              LIST(TERM-ID)
    END-EXEC.
 NEW-LINE-ATTRIBUTE.
    EXEC CICS
        ASKTIME
    END-EXEC.
    GO TO NEW-LINE-ATTRIBUTE.
    MOVE LOW-VALUES TO PRNTAREA.
    MOVE DFHBMPNL TO PRNTAREA.
```

Figure 18. Example of code containing a non-yielding loop

If you have a transaction that repeatedly abends with an abend code of AICA, first make sure the ICVR value has not been set too low. If the value seems reasonable, read “Investigating loops that cause transactions to abend with abend code AICA” on page 134 for advice on determining the limits of the loop.

If you have a stalled CICS region, diagnose the problem using the techniques in “What to do if CICS has stalled” on page 124. Check if the ICVR value has been set to zero. If it has, change the value and try to cause a transaction to abend with a code of AICA.

Yielding loops

Yielding loops are characterized by returning control at some point to a CICS routine that can suspend the looping task. However, the looping task is eventually resumed, and so the loop continues.

CICS is unable to use the runaway task timer to detect yielding loops, because the timer is reset whenever the task is suspended. Thus, the runaway task time is unlikely ever to be exceeded, and so the loop goes undetected by the system.

Yielding loops typically involve a number of programs. The programs might be linked to and returned from, or control might be transferred from one program to another in the loop. A yielding loop can also be confined to just one program, in which case it must contain at least one wait-enabling CICS command.

Figure 19 shows a yielding loop involving two programs, A and B. Program A links to program B, and then program B subsequently returns.

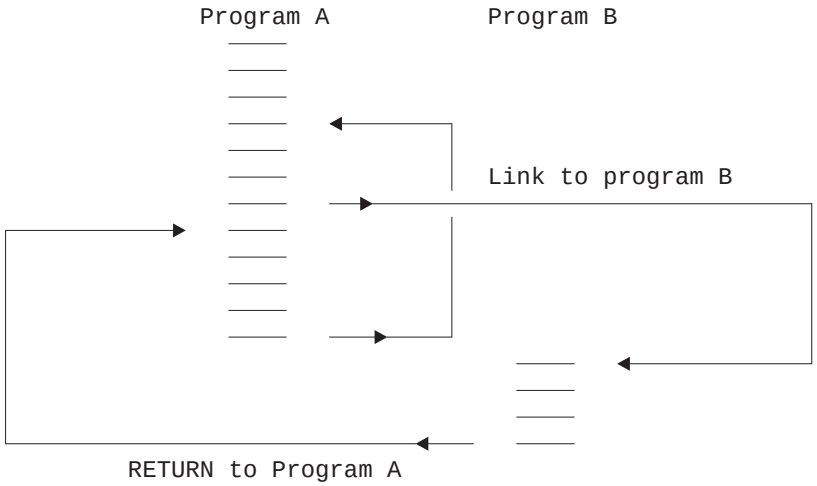


Figure 19. A yielding loop

Figure 20 shows a specific example of a yielding loop within a single program. This code issues the SUSPEND command, which is always a yielding type of command. Every time SUSPEND is issued, the dispatcher suspends the task issuing the request, and sees if any other task of higher priority can run. If no such task is ready, the program that issued the SUSPEND is resumed.

```
PROCEDURE DIVISION.
    EXEC CICS
        HANDLE CONDITION ERROR(ERROR-EXIT)
               ENDFILE(END-MSG)
    END-EXEC.
ROUTE-FILE.
    EXEC CICS
        ROUTE INTERVAL(0)
              LIST(TERM-ID)
    END-EXEC.
 NEW-LINE-ATTRIBUTE.
    EXEC CICS
        SUSPEND
    END-EXEC.
    GO TO NEW-LINE-ATTRIBUTE.
    MOVE LOW-VALUES TO PRNTAREA.
    MOVE DFHBMPNL TO PRNTAREA.
```

Figure 20. Example of code containing a yielding loop

You can detect a yielding loop only by circumstantial evidence such as repetitive output, or excessive use of storage. A fuller description of what to look out for is given in “Loops” on page 15.

If you suspect that you have a yielding loop, turn to “Investigating loops that are not detected by CICS” on page 136 for further guidance.

Investigating loops that cause transactions to abend with abend code AICA

If the loop causes a transaction to abend with abend code AICA, it must either be a tight loop or a non-yielding loop. You do not need to find which type you have, although this is likely to be revealed to you when you do your investigation.

Both a tight loop and a non-yielding loop are characterized by being confined to a single user program. You should know the identity of the transaction to which the program belongs, because it is the transaction that abended with code AICA when the runaway task was detected.

The documentation you need

When investigating loops that cause transactions to abend AICA, you need the CICS system dump accompanying the abend. System dumping must be enabled for dump code AICA.

You can use the system dump to find out:

- Whether the loop is in your user code or in CICS code
- If the loop is in your user code, the point at which the loop was entered

It is also useful to have trace running, as trace can help you to identify the point in your program where looping started. If you have a non-yielding loop, it can probably also show you some instructions in the loop.

A tight loop is unlikely to contain many instructions, and you might be able to capture all the evidence you need from the record of events in the internal trace table. A non-yielding loop may contain more instructions, depending on the EXEC CICS commands it contains, but you might still be able to capture the evidence you need from the record of events in the internal trace table. If you find that it is not big enough, direct tracing to the auxiliary trace destination instead.

Trace CICS system activity selectively, to ensure that most of the data you obtain is relevant to the problem. Set up the tracing like this:

1. Select level-1 special tracing for AP domain, and for the EXEC interface program (EI).
2. Select special tracing for just the task that has the loop, and disable tracing for all other tasks by turning the master system trace flag off.

You can find guidance about setting up these tracing options in Chapter 14, “Using traces in problem determination” on page 199.

Now start the task, and wait until it abends AICA. Format the CICS system dump with formatting keywords KE and TR, to get the kernel storage areas and the internal trace table. (See “Formatting system dumps” on page 244.) You now have the documentation you need to find the loop.

Looking at the evidence

Look first at the kernel task summary. The runaway task is flagged "*YES*" in the ERROR column. The status of the task is shown as "***Running**".

Now use the kernel task number for the looping task to find its linkage stack. If a user task is looping, DFHAPLI, a transaction manager program, should be near the top of the stack. You are likely to find other CICS modules at the top of the stack that have been invoked in response to the abend. For example, those associated with taking the dump. If you find any program or subroutine above DFHAPLI that has not been invoked in response to the error, it is possible that CICS code, or the code of another program, has been looping. If you find that the loop is within CICS code, contact the IBM Support Center. Make sure you keep the dump, because the Support Center staff need it to investigate the problem.

If the kernel linkage stack entries suggest that the loop is in your user program, you next need to identify the loop.

Identifying the loop

There are two approaches to identifying loops in user programs. You can use the trace table, or you can look in the transaction dump.

Using the trace table

Go to the last entry in the internal trace table, and work backward until you get to an entry for point ID AP 1942. The trace entry should have been made when recovery was entered after the transaction abended AICA. Make a note of the task number, so you can check that any other trace entries you read relate to the same abended task.

The entries preceding AP 1942 should have been made either just before the loop was entered (for a tight loop), or within the loop itself (for a non-yielding loop). Watch in particular for trace entries with the point ID AP 00E1. These are made on entry to the EXEC interface program (DFHEIP) whenever your program issues an EXEC CICS command, and again on exit from the EXEC interface program. Field B gives you the value of EIBFN, which identifies the specific command that was issued. (For a list of EIBFN values and their meanings, see the *CICS User's Handbook*.)

For trace entries made on exit from DFHEIP, field A gives you the response code from the request. Look carefully at any response codes—they could provide the clue to the loop. Has the program been designed to deal with every possible response from DFHEIP? Could the response code you see explain the loop?

If you see a repeating pattern of trace points for AP 00E1, you have a non-yielding loop. If you can match the repeating pattern to statements in the source code for your program, you have identified the limits of the loop.

If you see no repeating pattern of trace points for AP 00E1, it is likely that you have a tight loop. The last entry for AP 00E1 (if there is one) should have been made from a point just before the program entered the loop. You might be able to recognize the point in the program where the request was made, by matching trace entries with the source code of the program.

Using the transaction dump

First, find the PSW, and see if it points into your program. This is likely to be the case if you have a tight loop, and it should lead you to an instruction within the loop.

If the next instruction address is not within your code, it is of less value for locating the loop. However, you should attempt to identify the module containing the instruction, as it is likely to be one that was called during the execution of a CICS request made within the loop. Use the module index at the end of the formatted dump to find the module name. If the PSW address is not contained in one of these areas, another program was probably executing on behalf of CICS when the runaway task timer expired.

Note: It is possible that the loop was contained entirely within a module owned by CICS or some other product, and your program was not responsible for it at all. If you find that the loop is contained within CICS code, contact the IBM Support Center.

If the PSW does point to a module outside your application program, find the address of the return point in your program from the contents of register 14 in the appropriate register save area. The return address will lie within the loop, if the loop is not confined to system code.

When you have located a point within the loop, work through the source code and try to find the limits of the loop.

Finding the reason for the loop

When you have identified the limits of the loop, find the reason why the loop occurred. Assuming you have the trace, and EI level-1 tracing has been done, ensure that you can explain why each EIP entry is there. Verify that the responses are as expected.

A good place to look for clues to loops is immediately before the loop sequence, the first time it is entered. Occasionally, a request that results in an unexpected return code can trigger a loop. However, you usually can only see the last entry before the loop if you have CICS auxiliary trace running, because the internal trace table is likely to wrap before the AICA abend occurs.

Investigating loops that are not detected by CICS

You probably suspect that you have a loop through circumstantial evidence, and CICS has failed to detect it. You might, for example, have seen some sort of repetitive output, or statistics might have shown an excessive number of I/O operations or requests for storage. These types of symptom can indicate that you have a yielding loop.

The nature of the symptoms may indicate which transaction is involved, but you probably need to use trace to define the limits of the loop.

Use auxiliary trace to capture the trace entries, to ensure that the entire loop is captured in the trace data. If you use internal trace, there is a danger that wraparound will prevent you from seeing the whole loop.

Use the CETR transaction to set up the following tracing options. You can use the transaction dynamically, on the running CICS system. For guidance about using the CETR transaction, see Chapter 14, “Using traces in problem determination” on page 199.

1. Select level-1 special tracing for every component, using the CETR transaction. Capture as much trace information for the task as possible, because you do not yet know what functions are involved in the loop.
2. Set all standard tracing off, by setting the master system trace flag off.
3. Select special tracing for just the task containing the loop.
4. Set the auxiliary tracing status to STARTED, and the auxiliary switch status to ALL. As CETR allows you to control trace dynamically, you do not need to start tracing until the task is running and the symptoms of looping appear.

These steps ensure that you get all level-1 trace points traced for just the task you suspect of looping, the trace entries being sent to the auxiliary trace destination.

When you have captured the trace data, purge the looping task from the system. Use the CEMT INQUIRE TASK command to find the number of the task, and then purge it using either the CEMT SET TASK PURGE or the CEMT SET TASK FORCEPURGE command. This causes the transaction to abend, and to produce a transaction dump of the task storage areas.

Note: The use of FORCEPURGE is, in general, not recommended, because it can cause unpredictable system problems. For example, it causes task storage areas to be released, including I/O areas, without notifying any components that might be accessing them. If the FORCEPURGEd task was waiting for input, such an area might be written to after it is released. The storage might even be in use by another task when the input occurs.

The documentation you need

In addition to the auxiliary trace data and the transaction dump, you need source listings of all the programs in the transaction.

The trace data and the program listings should enable you to identify the limits of the loop. You need the transaction dump to examine the user storage for the program. The data you find there could provide the evidence you need to explain why the loop occurred.

Identifying the loop

Examine the trace table, and try to detect the repeating pattern of trace entries. If you cannot do so straightaway, remember that many different programs might be involved, and the loop could be large. Another possibility is that you might not have captured the entire loop in the trace data set. This could be because the loop did not have time to complete one cycle before you purged the transaction, or the trace data sets might have wrapped before the loop was complete.

Consider also the possibility that you might not be dealing with a loop, and the symptoms you saw are due to something else—poor application design, for example.

If you are able to detect a pattern, you should be able to identify the corresponding pattern of statements in your source code.

Note: The PSW is of no value in locating loops that are not detected by CICS. The contents of the PSW are unpredictable, and the PSW is not formatted in the transaction dump for ATCH abends.

Finding the reason for the loop

Look carefully at the statements contained in the loop. Does the logic of the code suggest why the loop occurred? If not, examine the contents of data fields in the task user storage. Look particularly for unexpected response codes, and null values when finite values are expected. Programs can react unpredictably when they encounter these conditions, unless they are tested for and handled accordingly.

What to do if you cannot find the reason for a loop

If you cannot find the reason for a non-yielding or a yielding loop using the techniques outlined above, there are two more approaches that you can adopt:

1. Use the interactive tools that CICS provides
2. Modify the program, and execute it again.

Investigating loops using interactive tools

If you have a non-yielding or a yielding loop, you can use the execution diagnostic facility (CEDF) to look at the various parts of your program and storage at each interaction with CICS. If you suspect that some unexpected return code might have caused the problem, CEDF is a convenient way of investigating the possibility.

CECI and CEBR are also useful for investigating loops. You can, for example, use them to examine the status of files and queues during the execution of your program. Programs can react unpredictably if records and queue entries are not found when these conditions are not tested for and handled accordingly.

Modifying your program to investigate the loop

If the program is extremely complex, or the data path difficult to follow, you may need to insert additional statements into the source code. Even extra ASKTIME commands allow you to use EDF and inspect the program at more points. You can also request dumps from within your program, and insert user trace entries, to help you to find the reason for the loop.

Chapter 8. Dealing with performance problems

When you have a performance problem, it might be characterized by one of the following symptoms, each of which represents a particular processing bottleneck. If so, turn directly to the relevant section:

1. Some tasks fail to get attached to the transaction manager—see “Finding why tasks fail to get attached to the transaction manager” on page 141.
2. Some tasks fail to get attached to the dispatcher—see “Finding why tasks fail to get attached to the dispatcher” on page 142.
3. Some tasks get attached to the dispatcher, but fail to get dispatched—see “Finding why tasks fail to get an initial dispatch” on page 144.
4. Tasks get attached to the dispatcher and then run and complete, but take a long time to do so—see “Finding out why tasks take a long time to complete” on page 146.

If you are only aware that performance is poor, and you have not yet found which of these is relevant to your system, read “Finding the bottleneck” on page 140.

There is a quick reference section at the end of this chapter that summarizes bottlenecks, symptoms, and actions that you should take.

© Copyright IBM Corp. 1979, 1999

Finding the bottleneck

Four potential bottlenecks can be identified for user tasks, and three for CICS system tasks. The bottlenecks are summarized in Figure 21.

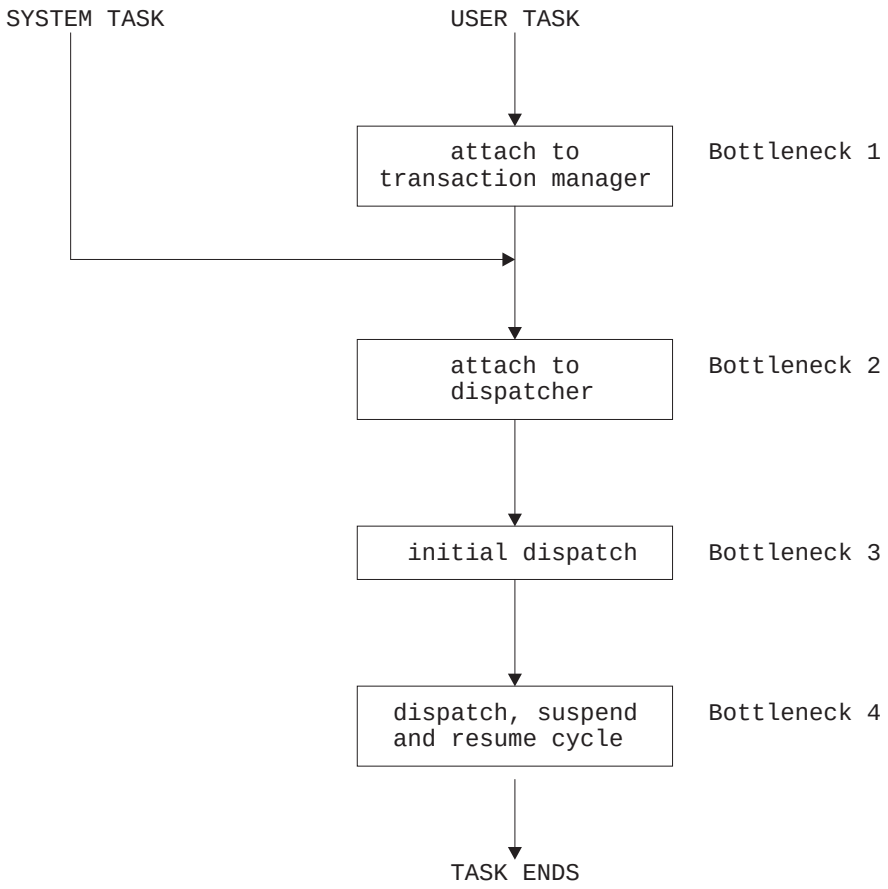


Figure 21. Potential performance bottlenecks in a CICS system

Each bottleneck is affected by a different set of system parameters, and you may find that adjusting the parameters solves the problem. It is useful to determine which bottleneck is causing your performance problem, so you can find out which parameters you need to consider.

If performance is particularly poor for any of the tasks in your system, you might be able to capture useful information about them with CEMT INQUIRE TASK. However, tasks usually run more quickly than you can inquire on them, even though there may be a performance problem. Then consider using performance class monitoring or tracing to get the information you need.

Initial attach to the transaction manager: If a task has not been attached to the transaction manager, you cannot get any information about its status online. CEMT INQUIRE TASK returns a response indicating that the task is not known. If the task has not already run and ended, this response means that it has not been attached to the transaction manager.

Guidance about finding out why tasks take a long time to get an initial attach to the transaction manager is given in “Finding why tasks fail to get attached to the transaction manager” on page 141.

Initial attach to the dispatcher: If a task has been attached to the transaction manager, but has not yet been attached to the dispatcher, CEMT INQUIRE TASK will show it to be 'SUSPENDED' on a resource type of MXT or TCLASS. These are the only valid reasons why a user task, having been attached to the transaction manager, would not be attached to the dispatcher.

If CEMT INQUIRE TASK returns anything other than this, the task is not waiting to be attached to the dispatcher. However, consider whether the MXT limit might be causing the performance problem, even though individual tasks are not being held up long enough for you to use CEMT INQUIRE TASK on them. In such a case, use monitoring and tracing to find just how long tasks are waiting to be attached to the dispatcher.

Guidance about finding whether the MXT limit is to blame for the performance problem is given in "Is the MXT limit preventing tasks from getting attached?" on page 142.

Initial dispatch: A task can be attached to the dispatcher, but then take a long time to get an initial dispatch. In such a case, CEMT INQUIRE TASK returns a status of 'READY' for the task. If you keep getting this response and the task fails to do anything, it is likely that the task you are inquiring on is not getting its first dispatch.

The delay might be too short for you to use CEMT INQUIRE TASK in this way, but still long enough to cause a performance problem. In such a case, use tracing or performance class monitoring for the task, either of which would tell you how long the task had to wait for an initial attachment to the dispatcher.

If you think your performance problem could be due to tasks taking a long time to get a first dispatch, read "Finding why tasks fail to get an initial dispatch" on page 144.

The dispatch, suspend, and resume cycle: If performance is poor and tasks are getting attached and dispatched, the problem lies with the dispatch, suspend and resume cycle. Tasks run, but the overall performance is poor. If you are able to show that tasks are getting attached and then dispatched, read "Finding out why tasks take a long time to complete" on page 146.

Finding why tasks fail to get attached to the transaction manager

A task might fail to get attached to the transaction manager for one of the following reasons:

1. The interval specified on an EXEC CICS START command might not have expired, or the time specified might not have been reached, or there might be some error affecting interval control.

   Guidance about investigating these possibilities is given in "Interval control waits" on page 115. Consider doing this only if INTERVAL or TIME was specified on the START command.

2. The terminal specified on an EXEC CICS START command might not be available. It could be currently OUTSERVICE, or executing some other task. Check its status using CEMT INQUIRE TERMINAL, and perhaps take some remedial action.

Remember that several tasks might be queued on the terminal, some of which might require operator interaction. In such a case, the transaction to be started might not get attached to the transaction manager for a considerable time.

3. A remote system specified on an EXEC CICS START command might not be available, or an error condition might have been detected in the remote system. In such a case, the error would not be reported back to the local system.

   Use CEMT INQUIRE TERMINAL to inquire on the status of the remote system.

Finding why tasks fail to get attached to the dispatcher

Two valid reasons why a user task might fail to get an initial attach to the dispatcher are[2]:

- The system is at the MXT limit (see "Is the MXT limit preventing tasks from getting attached?").
- The task belongs to a transaction class that is at its MAXACTIVE limit.

Is the MXT limit preventing tasks from getting attached?

Before the transaction manager can attach a user task to the dispatcher, the task must first qualify under the MXT (maximum tasks in the system) and transaction class limits. If a task is not getting attached, it is possible that one or both of these values is too small.

Try using CEMT INQUIRE TASK to show if a task is failing to get attached because of the MXT or transaction class limits. If you cannot use CEMT because the task is held up for too short a time, you can look at either the transaction global statistics, transaction class statistics, or the CICS performance-class monitoring records. Another option is to use CICS system tracing.

Using transaction manager statistics: To find out how often the MXT and transaction class limits are reached, look at the transaction global statistics and transaction class statistics. If you compare the number of times these limits are reached with the total number of transactions, you can see whether the values set for the limits are adversely affecting performance.

To gather statistics relating to the number of times that the MXT or transaction class limits are reached, you need to use, at the start of the run, the command CEMT PERFORM STATISTICS RECORD (or your site replacement) with the keywords TRANSACTION and TRANCLASS.

```
CEMT PERFORM STATISTICS RECORD [TRANCLASS TRANSACTION]
```

The statistics are gathered and recorded in the Data Management Facility (DMF) data set. You can format this data set by using the statistics utility program, DFHSTUP. Read the section on DFHSTUP in the *CICS Operations and Utilities Guide* for details on how to use this facility.

When formatting the DMF data set using DFHSTUP, you may find the following DFHSTUP control parameters useful:

---

[2] For a system task, there may not be enough storage to build the new task. This sort of problem is more likely to occur near peak system load times.

- SELECT APPLID=
- COLLECTION TYPE=REQ
- TIME START= ,STOP=
- DATE START= ,STOP=

See the section on DFHSTUP in the *CICS Operations and Utilities Guide* for details on how to code these parameters. If you code these control parameters correctly, you avoid the formatting of much information that may well be unnecessary at this point.

If MXT is never reached, or reached only infrequently, it is not affecting performance.

If MXT is reached for 5% of transactions, this might have a noticeable effect on performance. When the ratio reaches 10%, there is likely to be a significant effect on performance, and this could account for some tasks taking a long time to get a first attach.

Consider revising the MXT and transaction class values if the statistics indicate that they are affecting performance. For guidance about the performance considerations when you set these limits, see the *CICS Performance Guide*.

Using CICS monitoring: Use monitoring information to find out how long an individual task waits to be attached to the Dispatcher. If you want to summarize the monitoring data of particular transactions to assess the impact across many tasks, **and** you have a suitable MVS system that you can load your monitoring data into, use products such as Service Level Reporter (SLR).

Monitoring produces performance-class records (if performance-class monitoring is active) for each task that is executing or has executed in the CICS region. Performance-class records contain a breakdown of the delays incurred in dispatching a task, part of which is the impact on a task of the MXT limit and transaction class limits.

For further information on the data produced by CICS Monitoring see the *CICS Performance Guide*.

Using trace: Use trace if you want to find out just how long an individual task waits to be attached to the dispatcher.

If you do not want to do any other tracing, internal trace is probably a suitable destination for trace entries. Because the task you are interested in is almost inactive, very few trace entries are generated.

Select special tracing for the transaction associated with the task, and turn off all standard tracing by setting the master system trace flag off. Define as special trace points the level-1 trace points for transaction manager (XM), and for the CICS task controlling the facility that initiates the task, such as terminal control (TC). Make sure that no other trace points are defined as special. For guidance about setting up these tracing options, see Chapter 14, “Using traces in problem determination” on page 199.

When you have selected the options, start tracing to the internal trace table and attempt to initiate the task. When the task starts, get a system dump with CEMT

PERFORM SNAP. Format the dump using the keyword TR, to get the internal trace table.

Look for the trace entry showing terminal control calling the transaction manager with a request to attach the task, and the subsequent trace entry showing the transaction manager calling dispatcher domain with a request to attach the task. The time stamps on the two trace entries tell you the time that elapsed between the two events. That is equal to the time taken for the task to be attached.

Finding why tasks fail to get an initial dispatch

When a task is past the transaction class and MXT barriers, it can be attached to the dispatcher. It must then wait for its initial dispatch. If tasks are made to wait for a relatively long time for their first dispatch, you will probably notice the degradation in the performance of the system.

You can get evidence that tasks are waiting too long for a first dispatch from performance class monitoring. If you do find this to be the case, investigate the reasons for the delay. To calculate the initial dispatch delay incurred by a task use the following fields from the performance-class monitoring record:

DSPDELAY = First dispatch delay
TCLDELAY = Transaction Class delay
MXTDELAY = MXT delay

Using the above names:

Delay in dispatcher = DSPDELAY - (TCLDELAY + MXTDELAY)

If 'Delay in Dispatcher' is significantly greater than 0, the dispatcher could not dispatch the task immediately.

The factors that influence the length of time a task must wait before getting its first dispatch are:

- The priority of the task
- Whether the system is becoming short on storage

Priorities of tasks: Normally, the priorities of tasks determine the order in which they are dispatched. Priorities can have any value in the range 1–255. If your task is getting a first dispatch (and, possibly, subsequent dispatches) too slowly, you might consider changing its priority to a higher value.

You have no control over the priorities of CICS system tasks.

One other factor affecting the priorities of tasks is the priority aging multiplier, PRTYAGE, that you code in the system initialization parameters. This determines the rate at which tasks in the system can have their priorities aged. Altering the value of PRTYAGE affects the rate at which tasks are dispatched, and you probably need to experiment to find the best value for your system.

Storage stress conditions are detailed in the storage manager statistics. CICS attempts to alleviate the situation by releasing programs with no current user, and by not attaching new tasks. If these actions fail to eliminate storage stress or if the

SOS condition is caused by a suspended GETMAIN, one or both of these messages is sent to the console:

DFHSM0131 *applid* **CICS is under stress (short on storage below 16MB)**

DFHSM0133 *applid* **CICS is under stress (short on storage above 16MB)**

If you do not observe the SOS messages, you can find out how many times CICS has gone SOS from the storage manager statistics ('Times went short on storage'). You can also get this information from the storage manager domain DSA summary in a formatted system dump.

Note: Release of the storage cushion is not the only cause of CICS going SOS. The condition is also raised if a task makes an unconditional request for storage greater than the storage cushion size when the system is approaching SOS. In such a case, the cushion is not released, but the task making the unconditional request is suspended and message DFHSM0131I or DFHSM0133I may be issued. CICS resumes the suspended tasks immediately if storage is made available by CICS releasing unused programs. The short-on-storage condition remains until all the previously suspended tasks have obtained the storage they requested.

Two other conditions are recognized by the dispatcher on the approach to SOS, namely ‘storage getting short’ and ‘storage critical’. The two conditions affect the chance of new tasks getting a first dispatch.

From the ‘storage getting short’ point, through ‘storage critical’ and right up to SOS, the priorities of new user tasks are reduced in proportion to the severity of the condition[3]. At first, you are not likely to notice the effect, but as ‘storage critical’ is approached, new tasks might typically be delayed by up to a second before they are dispatched for the first time.

It is likely that ‘storage getting short’ and ‘storage critical’ occur many times for every occasion SOS is reached. If you want to see how often these points are reached, select level-2 tracing for the dispatcher domain and look out for trace point IDs **DS 0038** (‘storage getting short’) and **DS 0039** (‘storage critical’). Trace point **DS 0040** shows that storage is OK.

A summary of the effects of ‘storage getting short’, ‘storage critical’, and SOS is given in Table 24.

Table 24. How storage conditions affect new tasks getting started

State of storage	Effects on user tasks
Storage getting short	Priority of new user tasks reduced a little
Storage critical	Priority of new user tasks reduced considerably

[3] This sentence is untrue if the SIT parameter PRTYAGE is set to 0.

Finding out why tasks take a long time to complete

When a ready task is dispatched, it becomes a running task. It is unlikely to complete without being suspended at least once, and it is likely to go through the 'READY - RUNNING - SUSPENDED' cycle several times during its lifetime in the dispatcher.

The longer the task spends in the non-running state, either 'ready' or 'suspended', the greater your perception of performance degradation. In extreme cases, the task might spend so long in the non-running state that it is apparently waiting indefinitely. It is not likely to remain 'ready' indefinitely without running, but it could spend so long suspended that you would probably classify the problem as a wait.

The purpose of this section is to deal not with waiting tasks, but instead with tasks that complete more slowly than they should.

Here are some factors that can affect how long tasks take to complete:

- System loading
- Time-out interval for tasks
- Distribution of data sets on DASD volumes

Each of these factors is considered in turn.

The effect of system loading on performance

The most obvious factor affecting the time taken for a task to complete is system loading. For more information, see the *CICS Performance Guide*. Note in particular that there is a critical loading beyond which performance is degraded severely for only a small increase in transaction throughput.

The effect of task time-out interval on performance

The time-out interval is the length of time a task can wait on a resource before it is removed from the suspended state. A transaction that times out is normally abended.

Any task in the system can use resources and not allow other tasks to use them. Normally, a task with a large time-out interval is likely to hold on to resources longer than a task with a short time-out interval. Such a task has a greater chance of preventing other tasks from running. It follows that task time-out intervals should be chosen with care, to optimize the use of resources by all the tasks that need them.

A summary of performance bottlenecks, symptoms, and causes

Table 25 on page 147 contains a summary of potential performance bottlenecks, the symptoms you get if they are restricting the performance of your system, and the specific causes of the delays at each point.

Table 25. A summary of performance bottlenecks, symptoms and causes

Bottleneck	Symptoms	Possible causes
Initial attach to transaction manager	CEMT INQUIRE TASK does not know task. Tracing shows long wait for attach to transaction manager.	Interval on EXEC CICS START too long Terminal not available Remote system not available
Initial attach to dispatcher	CEMT INQUIRE TASK shows wait on MXT or transaction class. Tracing shows long wait for attach to dispatcher.	MXT or transaction class limits set too low
First dispatch	Performance class monitoring shows long wait for first dispatch. Storage statistics show CICS has gone SOS.	MXT or transaction class limits set too low Priority of task set too low Insufficient storage System under stress, or near it
SUSPEND / RESUME cycle	Tasks take a long time to complete.	System loading high Task time-out interval too large CICS data sets are susceptible to VSE LOCK activity

Chapter 9. Dealing with incorrect output

For the purpose of problem determination with this book, “incorrect output’ is explained in Chapter 2, “Classifying the problem” on page 9.

The following categories of incorrect output are dealt with in this chapter:

- “Trace output is incorrect”
- “Dump output is incorrect” on page 153
- “Wrong data has been displayed on a terminal” on page 156
- “Incorrect data is present on a VSAM data set” on page 163
- “An application did not work as expected” on page 163
- “Your transaction produced no output at all” on page 164
- “Your transaction produced some output, but it was wrong” on page 169

Trace output is incorrect

If you are unable to get the trace output you need, you can find guidance about solving the problem in this section. You can be very selective about the way CICS does tracing, and the options need to be considered carefully.

There are two main types of problem, explained below:

- Your tracing might have gone to the wrong destination.
- You might have captured the wrong data.

Tracing has gone to the wrong destination

CICS system trace entries go to the following destinations:

- CICS trace entries, other than exception traces, go to either of the following trace destinations if they are currently active:
  - The internal trace table
  - The current auxiliary trace data set

  You can inquire on the current destinations and set them to what you want using the CETR transaction.
- CICS exception traces go to the internal trace table even if internal tracing is not active, and to any other trace destination that is currently active.

Figure 39 on page 218 illustrates what you might see on a CETR screen, and indicates how you can change the options by overtyping the fields. From that illustration you can see that, from the options in effect, a normal trace call results in a trace entry being written to the auxiliary trace destination. If an exceptional condition occurred, the corresponding exception trace entry would be made both to the auxiliary trace data set and to the internal trace table, even though the internal trace status is STOPPED.

Note that the master system trace flag value only determines whether standard tracing is to be done for a task (see Figure 37 on page 213). It has no effect on any other tracing status.

Internal tracing goes to the internal trace table in main storage. The internal trace table is used as a buffer in which the trace entries are built regardless of the

© Copyright IBM Corp. 1979, 1999

destination. It, therefore, always contains the most recent trace entries, even if its status is STOPPED—if at least one of the other trace destinations is currently STARTED.

Auxiliary tracing goes to one of two data sets, if the auxiliary tracing status is STARTED. The current data set can be selected from the CETR screen by overtyping the appropriate field with A or B, as required. What happens when the data set becomes full is determined by the auxiliary switch status. Make sure that the switch status is correct for your system, or you might lose the trace entries you want, either because the data set is full or because they are overwritten.

The way in which trace entries are directed to the required destinations is illustrated in Figure 40 on page 219.

You have captured the wrong trace data

There are several ways in which you might capture the wrong trace data. The following are some sets of symptoms that suggest specific areas for attention:

1. You are not getting the right task tracing, because:
   - Tasks do not trace the right trace points for some components.
   - Transactions are not being traced when they are started from particular terminals.
   - There is no tracing for some terminals that interest you.

   If you are aware of symptoms like these, it is likely that you do not have the right task tracing options set up. Turn to “You are not getting the correct task tracing” on page 151 for further guidance.
2. You are getting the wrong amount of data traced, because:
   - Tracing is not being done for all the components you want, so you are getting too little information.
   - Tracing is being done for too many components, so you are getting more information than you want.
   - You are not getting the right trace points (level-1 or level-2) traced for some of the components.
   - Tasks are not tracing the component trace points you want. This evidence suggests CICS component tracing selectivity is at fault.

   If your observations fit any of these descriptions, turn to “You are not getting the correct component tracing” on page 151 for guidance about fixing the problem.
3. The data you want is missing entirely from the trace table.

   If you have this sort of problem, turn to “The entries you want are missing from the trace table” on page 151 for guidance about finding the cause.

It is worth remembering that the more precisely you can define the trace data you need for any sort of problem determination, the more quickly you are likely to get to the cause of the problem.

You are not getting the correct task tracing

If you are not getting the correct task tracing, use the CETR transaction to check the transaction and terminal tracing options, and if necessary change them.

You can define whether you want standard or special tracing for specific transactions, and standard or special tracing for transactions started at specific terminals. You can also suppress tracing for transactions and terminals that do not interest you. The type of task tracing that you get (standard or special) depends on the type of tracing for the corresponding transaction-terminal pair, in the way shown in Figure 35 on page 212.

You can deduce from the table that it is possible to get standard tracing when a transaction is initiated at one terminal, and special tracing when it is initiated from another terminal. This raises the possibility of setting up inappropriate task tracing options, so the trace entries that interest you—for example, when the transaction is initiated from a particular terminal—are not made.

You are not getting the correct component tracing

If you are not getting the correct component tracing, use the CETR transaction to inquire on the current component tracing options and, if necessary, to change them.

First, check that you are only tracing components that interest you. If some other components are being traced, change the options so they are no longer traced for standard tracing or for special tracing, as appropriate.

Next, check that the right tracing levels have been defined for standard tracing and special tracing. Remember that, whenever a task that has standard tracing is running, the trace points that you have defined as standard for a component are traced whenever that component is invoked. Similarly, special trace points are traced whenever special task tracing is being done.

Figure 37 on page 213 illustrates the logic used to determine whether a trace call is to be made from a trace point.

If you are satisfied that the component tracing selectivity is correct but you are still getting too much or too little data, read “You are not getting the correct task tracing” above.

The entries you want are missing from the trace table

Read this section if one or more entries you were expecting were missing entirely from the trace table.

There are several possible situations:

1. The trace entry did not appear at the expected time:
   - If tracing for some components or some tasks did not appear, you might not have set up the tracing selectivity correctly. For guidance about checking and correcting the options, refer to “You are not getting the correct task tracing” and “You are not getting the correct component tracing.”
   - If you attempted to format the auxiliary trace data set selectively by transaction or terminal, and trace entries for the transaction or terminal were missing entirely, it could be that you did not capture the

corresponding “transaction attach” (point ID AP F004, KC level-1) trace entry.

When you select trace entries by specifying TRANID or TERMID parameters in the DFHTU410 trace control statements, DFHTU410 searches for any transaction attach trace entries that contain the specified TRANID or TERMID. It then formats any associated trace entries, identified by the TASKID found in the transaction attach trace entry data.

It follows that you must have KC level-1 tracing selected for the task in question at the time it is attached if you want to format the auxiliary trace data set selectively by transaction or terminal.

For more details about trace formatting using DFHTU410, see the *CICS Operations and Utilities Guide*.

If the options were correct and tracing was running at the right time, but the trace entries you wanted did not appear, it is likely that the task you were interested in did not run or did not invoke the CICS components you expected. Examine the trace carefully in the region in which you expected the task to appear, and attempt to find why it was not invoked. Remember also that the task tracing options might not, after all, have been appropriate.

2. The earliest trace entry was later than the event that interested you.

   If tracing was running at the right time, it is likely that the trace table wrapped round and earlier entries were overwritten.

   Internal trace always wraps when it is full. Try using a bigger trace table, or direct the trace entries to the auxiliary trace destination.

   Note: Changing the size of the internal trace table during a run causes the data that was already there to be destroyed. In such a case, the earliest data would have been recorded after the time when you redefined the table size.

   Auxiliary trace switches from one data set to the next when it is full, if the autoswitch status is NEXT or ALL:

   - If the autoswitch status is NEXT, the two data sets can fill up but earlier data cannot be overwritten. Your missing data might be in the initial data set, or the events you were interested in might have occurred after the data sets were full. In the second case, you can try increasing the size of the auxiliary trace data sets.
   - If the autoswitch status is ALL, you might have overwritten the data you wanted. The initial data set is reused when the second extent is full. Try increasing the size of the auxiliary trace data sets.

3. You cannot find an exception trace entry that you expected.

   Bear in mind that exception tracing is always done to the internal trace table irrespective of the status of any other type of tracing. So, if you missed it in your selected trace destination, try looking in the internal trace table.

Dump output is incorrect

Read this section if you did not get the dump output you were expecting.

The things that can go wrong are:

- The dump you got did not seem to relate to the CICS partition in which you were interested.
- You did not get a dump when an abend occurred.
- Some dump IDs were missing from the sequence of dumps in the dump data set.
- You did not get the correct data formatted from a system dump.

The sections that follow give guidance about resolving the possible problems you may encounter.

The dump you got did not seem to relate to your CICS partition

If you have experienced this problem, it is likely that you have dumped the wrong CICS partition. It should not occur if you are running a single partition. If you invoked the dump from the CICS master terminal using the CEMT PERFORM SNAP transaction, check that you were using the master terminal for the correct CICS partition. This is more likely to be a problem if you have a VTAM network, because that allows you to switch a single physical VTAM terminal between the different CICS partitions.

You did not get a dump when an abend occurred

Read this section if you have experienced any of these problems:

- A transaction abended, but you did not get a transaction dump.
- A transaction abended and you got a transaction dump, but you did not get the system dump you wanted at the same time.
- A system abend occurred, but you did not get a system dump.

There are, in general, two reasons why dumps might not be taken:

- Dumping was suppressed because of the way the dumping requirements for the CICS partition were defined. The valid ways that dumping can be suppressed are described in detail in the sections that follow.
- A system error could have prevented a dump from being taken. Some of the possibilities are:
  - No transaction or system dump data sets were available.
  - An I/O error occurred on a transaction or a system dump data set.
  - There was insufficient space to write the dump in the dump data set. In such a case, you might have obtained a partial dump.

    Depending on the areas that are missing from the dump, the dump formatting program might subsequently be able to format the data that is there, or it might not be able to format the data at all.

  For each of these system errors, there should be a message explaining what has happened. See the *VSE/ESA Messages and Codes Volume 3* manual for guidance about the action to take.

How dumping can be suppressed

If you did not get a dump when an abend occurred, and there was no system error, the dumping that you required must somehow have been suppressed. There are several levels at which dumping can be suppressed:

Find out which type of dump suppression applies to your system before you decide what remedial action to take.

Global suppression of system dumping

System dumping can be suppressed globally in two ways:

- By coding the DUMP=NO system initialization parameter
- By using the CEMT command CEMT SET SYSTEM DUMPING (NOSYSDUMP)
- By using the system programming command EXEC CICS SET SYSTEM DUMPING, with a CVDA value of NOSYSDUMP

If system dumping has been suppressed globally by any of these means, any system dumping requirements specified in the transaction dump table and the system dump table are overridden.

Inquire whether system dumping has been suppressed globally by using the CEMT or EXEC CICS INQUIRE SYSTEM DUMPING commands. If necessary, cancel the global suppression of system dumping using CEMT or EXEC CICS SET SYSTEM DUMPING (SYSDUMP).

Suppression of system dumping from a global user exit program

System dumping can be suppressed for specific dump codes by an XDUREQ user exit program. For programming information about the XDUREQ global user exit program, see the *CICS Customization Guide*.

If an exit program that suppresses system dumping for a particular dump code is enabled, system dumping is not carried out for that dump code. This overrides any system dumping requirement specified for the dump code in the dump table.

The exit program can suppress system dumps only while it is enabled. If you want the system dumping suppression to be canceled, issue an EXEC CICS DISABLE command for the program. Any system dumping requirements specified in the dump table then take effect.

Suppression of dumping for individual transactions

For individual transactions, you can suppress transaction dumps by using the EXEC CICS SET TRANSACTION DUMPING system programming command, or by using the DUMP attribute on the RDO TRANSACTION resource definition. None of the dumping requirements specified in the transaction dump table are met if a transaction abends for which dumping is suppressed.

Use EXEC CICS INQUIRE TRANSACTION DUMPING to see whether dumping has been suppressed for a transaction, and then use the corresponding SET command to cancel the suppression if necessary.

Suppression of dumping by dump table options

If transaction dumping and system dumping are not suppressed by any of the preceding mechanisms, the dump table options determine whether or not you get a dump for a particular dump code. For details, see Chapter 15, “Using dumps in problem determination” on page 225.

You can inquire on transaction and system dump code attributes using CEMT INQUIRE TRDUMPCODE and CEMT INQUIRE SYDUMPCODE, respectively. You must specify the dump code you are inquiring on.

If you find that the dumping options are not what you want, use CEMT SET TRDUMPCODE or CEMT SET SYDUMPCODE to change the values of the attributes accordingly.

- **If you had no transaction dump when a transaction abended**, use CEMT INQUIRE TRDUMPCODE and check if an attribute of TRANDUMP or NOTRANDUMP is specified for this dump code. The attribute needs to be TRANDUMP if a transaction dump is to be taken.

  If the attribute is shown to be TRANDUMP, look next at the maximum number of dumps specified for this dump code, and compare it with the current number. The values are probably equal, showing that the maximum number of dumps have already been taken.

- **If you had a transaction dump but no system dump**, use CEMT INQUIRE TRDUMPCODE and check whether there is an attribute of SYSDUMP or NOSYSDUMP for the dump code. You must SYSDUMP specified if you are to get a system dump as well as the transaction dump.

  Check also that you have not had all the dumps for this dump code, by comparing the maximum and current dump values.

- **If you had no system dump when a system abend occurred**, use CEMT INQUIRE SYDUMPCODE and check whether you have an attribute of SYSDUMP or NOSYSDUMP for the dump code. You need SYSDUMP if you are to get a system dump for this type of abend.

  Finally, check the maximum and current dump values. If they are the same, you need to reset the current value to zero.

Some dump IDs were missing from the sequence of dumps

CICS keeps a count of the number of times that dumping is invoked during the current run, and the count is included as part of the dump ID given at the start of the dump.

Note: SDUMPs produced by the kernel do not use the standard dump domain mechanisms, and always have a dump ID of 0/0000.

If both a transaction dump and a system dump are taken in response to the event that invoked dumping, the same dump ID is given to both. However, if just a transaction dump or just a system dump is taken, the dump ID is unique to that dump.

The complete range of dump IDs for any run of CICS is, therefore, distributed between the set of system dumps and the set of transaction dumps, but neither set of dumps has them all.

Figure 22 on page 156 gives an example of the sort of distribution of dump IDs that might occur. Note that each dump ID is prefixed by the run number, in this case 23, and that this is the same for any dump produced during that run. This does not apply to SDUMPs produced by the kernel; these **always** have a dump ID of 0/0000.

On system dump data set	On transaction dump data set
ID=23/0001 ID=23/0002 ID=23/0004 ID=23/0006 ID=23/0007	 ID=23/0002 ID=23/0003 ID=23/0005 ID=23/0008

Figure 22. Typical distribution of dump IDs between dump data sets

For further discussion of the way CICS manages transaction and system dumps, see Chapter 15, “Using dumps in problem determination” on page 225.

You did not get the correct data formatted from a CICS system dump

If you did not get the correct data formatted from a CICS system dump, these are the most likely explanations:

- You did not use the correct dump formatting keywords. If you do not specify any formatting keywords, the whole system dump is formatted. However, if you specify any keywords at all, you must be careful to specify keywords for *all* the functional areas you are interested in.
- You used the correct dump formatting keywords, but the dump formatting program was unable to format the dump correctly because it detected an error. In such a case, you should be able to find a diagnostic error message from the dump formatter.
- A partial dump might have been taken as the VSE SYSDUMP library is full.

Wrong data has been displayed on a terminal

There are many reasons why you might get the wrong data displayed, some with system-related causes and some with application-related causes. If you think that it is system-related, read this section for some suggestions on likely areas in which to start your investigations.

For the present purpose, a terminal is considered to be any device where data can be displayed. It might be some unit with a screen, or it could be a printer. Many other types of terminals are recognized by CICS, including remote CICS regions, and batch partitions, but they are not considered in this section on incorrect output.

Broadly, there are two types of incorrect output that you might get on a screen, or on a printer:

- The data information is wrong, so unexpected values appear on the screen or in the hard copy from a printer.

- The layout is incorrect on the screen or in the hard copy. That is, the data is formatted wrongly.

In practice, you may sometimes find it difficult to distinguish between incorrect data information and incorrect formatting. In fact, you seldom need to make this classification when you are debugging this type of problem.

Sometimes, you might find that a transaction runs satisfactorily at one terminal, but fails to give the correct output on another. This is probably due to the different characteristics of the different terminals, and you should find the answer to the problem in the sections that follow.

The preliminary information you need to get

Before you can investigate the reasons why incorrect output is displayed at a terminal, you need to gather some information about the transaction running at the terminal, and the about terminal itself.

The first things you need to know are:

- The identity of the transaction associated with the incorrect output.
- For an autoinstalled terminal, the model number, to ensure that you inquire on the correct RDO TYPETERM resource definition. You can find this from the autoinstall message in the CADL log.

Depending on the symptoms you have experienced, you probably need to examine the RDO PROFILE definitions for the transaction, and the RDO TYPETERM resource definitions for the affected terminal. The attributes most likely to be of interest are SCRNSIZE for the RDO PROFILE, and ALTSCREEN, ALTPAGE, PAGESIZE, EXTENDEDDS, and QUERY for RDO TYPETERM. Other attributes might also be significant, but the values you find for the attributes named here can often explain why the incorrect output was obtained.

Specific types of incorrect output, and their possible causes

This section contains some suggestions about what to do for specific types of incorrect output, and what might be at fault.

Logon rejection message

If you get a logon rejection message when you attempt to log on to CICS, it could be that the RDO TYPETERM resource definitions for the terminal are incorrect. A message recording the failure is written to the CSNE log or, in the case of autoinstall, to the CADL log.

You are likely to get a logon rejection if you attempt to specify anything other than QUERY(NO) for a terminal that does not have the structured query field feature. Note that NO is the default value for RDO TYPETERM resource definitions that you supply, but YES is the value for RDO TYPETERM resource definitions that are supplied with CICS. If you have a persistent problem with logon rejection, you can use the VTAM buffer trace to find out more about the reasons for the failure.

Unexpected messages and codes

If the “wrong data” is in the form of a message or code that you do not understand, look in the appropriate manual for an explanation of what it means.

Messages that are prefixed by DFH originate from CICS— look in the *VSE/ESA Messages and Codes Volume 3* manual for these. For codes that appear in the space at the bottom of the screen where status information is displayed, look in the appropriate guide for the terminal.

The following are examples of common errors that can cause messages or codes to be displayed:

- SCRNSIZE(ALTERNATE) has been specified in an RDO PROFILE resource definition, and too many rows have been specified for ALTSCREEN and ALTPAGE in the RDO TYPETERM resource definition for the terminal.
- An application has sent a spurious hex value corresponding to a control character in a data stream. For example, X'11' is understood as “set buffer address” by a 3270 terminal, and the values that follow are interpreted as the new buffer address. This eventually causes an error code to be displayed.

  If you suspect this may be the cause of the problem, check your application code carefully to make sure it cannot send any unintended control characters.
- EXTENDEDDS(YES) has been specified in the RDO TYPETERM resource definition for a device that does not support this feature. In such a case, a message is sent to the screen, and a message might also be written to the CSMT log.

  The default value for EXTENDEDDS is NO, but check to make sure that YES has not been specified if you know your terminal is not an extended data stream device.

Unexpected appearance of uppercase or lowercase characters

If the data displayed on your terminal has unexpectedly been translated into uppercase characters, or if you have some lowercase characters when you were expecting uppercase translation, you need to look at the options governing the translation.

These are the significant properties of the various translation options you have:

- The ASIS option on EXEC CICS BMS and terminal control commands specifies that lowercase characters in an input data stream are not to be translated to uppercase.

  ASIS overrides the UCTRAN attributes for both RDO TYPETERM and PROFILE resource definitions.
- The UCTRAN attribute of the RDO TYPETERM resource definition states whether lowercase characters in 3270 input data streams are to be translated to uppercase for terminals with this TYPETERM definition.

  The UCTRAN attribute of RDO TYPETERM resource definition is overridden by ASIS, but it overrides the UCTRAN attribute of an RDO PROFILE resource definition.
- The UCTRAN attribute of an RDO PROFILE resource definition states whether lowercase characters in the input data stream are to be translated to uppercase

for transactions with this PROFILE running on VTAM terminals. The PROFILE UCTRAN value is valid only for VTAM terminals.

The UCTRAN option for an RDO PROFILE resource definition is overridden by both the UCTRAN option for an RDO TYPETERM resource definition and the ASIS option on EXEC CICS BMS and terminal control commands.

- If the ASIS option is NOT specified, then if either the PROFILE or the TYPETERM resource definitions specify UCTRAN(YES), the data presented to the transaction is translated.

Note: User exit XZCIN can also be used to perform uppercase translation.

The UPPERCASE option in the offline utilities (DFHDU410, DFHSTUP, DFHTU410) specify whether all lowercase characters are to be translated to uppercase characters.

Figure 23 and Figure 24 summarize whether or not you get uppercase translation, depending on the values of these options.

ASIS option not specified		TYPETERM	
		UCTRAN(YES)	UCTRAN(NO)
PROFILE	UCTRAN(YES)	YES	YES
	UCTRAN(NO)	YES	NO

Figure 23. Uppercase translation truth table—ASIS option not specified

ASIS option is specified		TYPETERM	
		UCTRAN(YES)	UCTRAN(NO)
PROFILE	UCTRAN(YES)	NO	NO
	UCTRAN(NO)	NO	NO

Figure 24. Uppercase translation truth table—ASIS option is specified

The UPPERCASE option in the offline utilities (DFHDU410, DFHSTUP, and DFHTU410) specifies whether all lowercase characters are to be translated to uppercase.

CRTE and uppercase translation

Initiating a CRTE session: The input required to start a CRTE routing session is of the form:

```
CRTE SYSID(xxxx),TRPROF(yyyyyyyy)
```

Translation to uppercase is dictated by the TYPETERM of the terminal at which CRTE was entered and CRTE's transaction profile definition as shown in Table 26.

Table 26. Uppercase translation on CRTE session initiation

TYPETERM UCTRAN	CRTE PROFILE UCTRAN	INPUT TRANSLATED TO UPPERCASE
YES	YES/NO	ALL OF THE INPUT
NO	NO	NONE OF THE INPUT. See note.
NO	YES	ALL OF THE INPUT EXCEPT THE TRANSID. See note.
TRANID	YES	ALL OF THE INPUT
TRANID	NO	TRANSID ONLY

Note: Note that if the transid CRTE is not entered in upper case, it will not be recognized (unless there is a lower/mixed case alias), and message DFHAC2001 will be issued.

Input within the CRTE session: During the CRTE routing session, uppercase translation is dictated by the TYPETERM of the terminal at which CRTE was initiated and the transaction profile definition of the transaction being initiated (which has to be a valid transaction on the application owning region) as shown in Table 27.

Table 27. Uppercase translation during CRTE session

TYPETERM UCTRAN	TRANSACTION PROFILE (AOR) UCTRAN	INPUT TRANSLATED TO UPPERCASE
YES	YES/NO	ALL OF THE INPUT
NO	NO	NONE OF THE INPUT. See note.
NO	YES	ALL OF THE INPUT EXCEPT THE TRANSID. See note.
TRANID	YES	ALL OF THE INPUT
TRANID	NO	TRANSID ONLY

Note: Note that if the transid CRTE is not entered in upper case, it will not be recognized (unless there is a lower/mixed case alias defined on the AOR) and message DFHAC2001 will be issued.

During a CRTE routing session, if the first six characters entered at a screen are `CANCEL`, CICS will recognize this input in upper, lower or mixed case and end the routing session.

For more information on the ALIAS attribute of the RDO TRANSACTION resource definition, see the 'Transaction' section of the *CICS Resource Definition Guide*.

Be aware that when transaction routing from CICS Transaction Server for VSE/ESA Release 1 to an earlier release that does not support transaction based uppercase translation, uppercase translation only occurs if it is specified in the RDO TYPETERM.

EXEC CICS SET TERMINAL and uppercase translation

In a single system, if the EXEC CICS SET TERMINAL command is issued for a terminal while it is running a transaction performing RECEIVE processing, unpredictable results may occur. This is because the command can override the RDO TYPETERM definition regarding uppercase translation and RECEIVE processing interrogates the uppercase translate status of the terminal in order to establish whether translation is required.

In a transaction routing environment, the system programmer who issues the EXEC CICS SET TERMINAL command should be aware (for VTAM terminals) that the TOR terminal uppercase translate status is copied to the AOR surrogate terminal on every flow across the link from the TOR to the AOR. Consequently:

- The EXEC CICS SET TERMINAL change of uppercase translate status will only take effect on the AOR on the next flow across the link.
- Any AOR TYPETERM definition used to hard code remote terminal definitions will be overridden with the TOR values for uppercase translate status.
- EXEC CICS INQUIRE TERMINAL issued on the AOR can return misleading uppercase translation status of the terminal, since the correct status on the TOR may not yet have been copied to the AOR.
- The processing of RECEIVE requests on the TOR and AOR can interrogate the uppercase translate status of the terminal. Therefore unpredictable results can also occur if the system programmer issues the EXEC CICS SET TERMINAL command during receive processing.

Katakana terminals—mixed English and Katakana characters

If you are using a Katakana terminal, you might see some messages containing mixed English and Katakana characters. That is because Katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as Katakana characters. If you have any Katakana terminals connected to your CICS system, specify the MSGCASE=UPPER system initialization parameter to ensure that messages contain uppercase characters only.

The offline utilities DFHDU410, DFHSTUP and DFHTU410 have an extra parameter to ensure all output is translated to uppercase. See the *CICS Operations and Utilities Guide* for details on how to use these parameters.

Wrong data values are displayed

If the data values are wrong on the user's part of the screen (the space above the area used to display status information to the operator), or in the hard copy produced by a printer, it is likely that the application is at fault.

Some data is not displayed

If you find that some data is not being displayed, consider these possibilities:

- The SENDSIZE value for the RDO TYPETERM resource definition could be too large for the device receiving the data. Its receiving buffer could then overflow, with some data being lost.
- SCRNSIZE(ALTERNATE) might be specified in the RDO PROFILE resource definition for the transaction running at the terminal, while default values for ALTSCREEN and ALTPAGE are allowed in the RDO TYPETERM resource definition for the terminal.

  The default values for ALTSCREEN and ALTPAGE are 0 rows and 0 columns, so no data could then be displayed if SCRNSIZE(ALTERNATE) were specified.
- EXTENDEDDS(YES) is specified (on the RDO TYPETERM resource definition) for a device that does not support this feature.

Early data is overlaid by later data

Early data can be overlaid by later data, so that data appears in the wrong order, when the SENDSIZE value of the RDO TYPETERM resource definition is too large for the device receiving the data. This is because the buffer can wrap when it is full, with the surplus data overlaying the first data that was received.

The data is formatted wrongly

Incorrect formatting of data can have a wide range of causes, but here are some suggestions of areas that can sometimes be troublesome:

- BMS maps are incorrect.
- Applications have not been recompiled with the latest maps.
- Different numbers of columns have been specified for ALTSCREEN and ALTPAGE in the RDO TYPETERM resource definitions for the terminal. This can lead to unpredictable formatting errors. However, you will not see them unless SCRNSIZE(ALTERNATE) has been specified in the RDO PROFILE resource definitions for the transaction running at the terminal.
- The PAGESIZE values included in the RDO TYPETERM resource definitions must suit the characteristics of the terminal, or you get formatting errors.

  For a screen display, the number of columns specified must be less than or equal to the line width. For a printer, the number of columns specified must be less than the line width, or else both BMS (if you are using it) and the printer might provide a new line, giving you extra spacing you do not want.

  The default values for PAGESIZE depend on the value you specify for the DEVICE keyword.
- If you get extra line feeds and form feeds on your printer, it could be that an application is sending control characters that are not required because the printer is already providing end of line and end of form operations.

  If your application is handling the buffering of output to a printer, make sure that an “end of message” control character is sent at the end of every buffer full of data. Otherwise, the printer might put the next data it receives on a new line.

Tools for debugging terminal output in a VTAM environment

Among the debugging tools you have, the most likely to be of particular use for investigating terminal incorrect output errors in a VTAM environment is VTAM buffer trace. This is a function of VTAM itself, and you need to read the appropriate manual in the VTAM library to find out how to use it.

For a description of the use of these two types of tracing in CICS problem determination, see Chapter 14, “Using traces in problem determination” on page 199.

Incorrect data is present on a VSAM data set

If EXEC CICS READ UPDATE is not used, an error can occur because VSAM allows a record to be read by one transaction while another transaction is updating it.

If the first transaction were to take some action based on the value of the record, the action would probably be erroneous.

For example, in inventory control, a warehouse has 150 items in stock. 100 items are sold to a customer, who is promised delivery within 24 hours. The invoice is prepared, and this causes a transaction to be invoked that is designed to read the inventory record from a VSAM data set and updates it accordingly.

In the meantime, a second customer also asks for 100 items. The salesperson uses a terminal to inquire on the number currently in stock. The “inquire” transaction reads the record that has been read for update but not yet rewritten, and returns the information that there are 150 items. This customer, too, is promised delivery within 24 hours.

Errors of this kind are prevented by the use of EXEC CICS READ UPDATE.

An application did not work as expected

It is not possible to give specific advice on dealing with this sort of problem, but the points and techniques that follow should help you to find the area where the failure is occurring.

General points for you to consider

1. Make sure you can define exactly what happened, and how this differs from what you expected to happen.
2. Check the commands you are using for accuracy and completeness. For programming information about EXEC CICS commands, see the *CICS Application Programming Reference* and *CICS System Programming Reference* manuals. Are any default values the ones you really want? Does the description of the effect of each command match your expectations?
3. Can you identify a failing sequence of commands? If so, can it be reproduced using CECI?
4. Consider the resources required by the application. Are they defined as expected?
5. Are the required functions in the failing functional area available in this system?

6. For “input” type requests, does the item exist? You can verify this using offline utilities.
7. For “output” type requests, is the item created? Verify that the before and after images are as expected.

Using traces and dumps

Traces and dumps can give you valuable information about unusual conditions that might be causing your application to work in an unexpected way.

1. If the path through the transaction is indeterminate, insert user trace entries at all the principal points.
2. If you know the point in the code where the failure occurs, insert a CICS system dump request immediately after it.
3. Use CETR to select special tracing for the level-1 trace points for all components. Select special tracing for the failing task only, and disable all standard tracing by setting the master system trace flag off.
4. Run the transaction after setting the trace options, and wait until the system dump request is executed. Format the internal trace table from the dump (formatting keyword TR), and examine the trace entries before the failure. Look in particular for unusual or unexpected conditions, possibly ones that the application is not designed to handle.

Your transaction produced no output at all

If your transaction produced no output at all, you need to carry out some preliminary checks before looking at the problem in detail. You might be able to find a simple explanation for the failure.

Are there any messages explaining why there is no output?

Look carefully in each of the transient data destinations CSMT, and CSTL for any messages that might relate to the task. You could find one there that explains why you received no output.

If you can find no such message, the next step is to get some information about the status of the transaction that produced no output, your terminal, and the CICS system.

Can you use the terminal where the transaction should have started?

Go to the terminal where the transaction should have started, and note whether the keyboard is locked. If it is, press RESET. Now try issuing CEMT INQUIRE TASK (or your site replacement) from the terminal.

If you cannot issue CEMT INQUIRE TASK from the terminal, one of these explanations applies:

- The task that produced no output is still attached to the terminal.
- The terminal where you made the inquiry is not in service.
- There is a system-wide problem.
- You are not authorized to use the CEMT transaction. (This may be because you have not signed on to the terminal and the CEMT transaction is not

authorized for that terminal. If you ***have*** signed on to the terminal, you are probably authorized to use CEMT.)

Try to find a terminal where you can issue CEMT INQUIRE TASK. If no terminal seems to work, there is probably a system-wide problem. Otherwise, see if the task you are investigating is shown in the summary.

- If the task is shown, it is probably still attached, and either looping or waiting. Turn to "No output—what to do if the task is still in the system" to see what to do next.
- If the task is not shown, there is a problem with the terminal where you first attempted to issue CEMT INQUIRE TASK.

If you are able to issue CEMT INQUIRE TASK from the terminal where the transaction was attached, one of these explanations applies:

- The transaction gave no output because it never started.
- The transaction ran without producing any output, and terminated.
- The transaction started at another terminal, and might still be in the system. If it is still in the system, you can see it in the task summary that you got for CEMT INQUIRE TASK. It is probably looping or waiting. Turn to "No output—what to do if the task is still in the system" for advice about what to do next. If you do not see the task in the summary, turn to "No output—what to do if the task is not in the system."

No output—what to do if the task is still in the system

If you obtained no output and the task is still in the system, it is either waiting for a resource, or looping. You should get an indication of which of these two conditions is the most likely from the status for the task returned by CEMT INQUIRE TASK.

You have a suspended task, treat this as a "wait" problem. Use the techniques of Chapter 6, "Dealing with waits" on page 51 to investigate it further.

You have a running task, it is likely to be looping. Turn to Chapter 7, "Dealing with loops" on page 129 to find out what to do next.

No output—what to do if the task is not in the system

If you have obtained no output and CEMT INQUIRE TASK shows the task is not in the system, one of two things could have happened:

- Your transaction never started.
- Your transaction ran, but produced no output.

  Note: If you are not getting output on a printer, the reason could be simply that you are not setting on the START PRINTER bit in the write control character. You need to set this bit to get printed output if you have specified the STRFIELD option on a CONVERSE or SEND command, which means that the data area specified in the FROM option contains structured fields. Your application must set up the contents of the structured fields.

Your task might have been initiated by direct request from a terminal, or by automatic task initiation (ATI). Most of the techniques apply to both sorts of task, but there are some extra things to investigate for ATI tasks. Carry out the tests which apply to all tasks first, then go on to the tests for ATI tasks if you need to.

Did the task run? Techniques for all tasks

There are many different techniques for finding out if a transaction started, or if it ran but produced no output. Use the ones that are most convenient at your installation.

Using CICS system trace entry points:

CICS system tracing is probably the most powerful technique for finding out whether a transaction ever started. You might need to direct the trace output to the auxiliary trace destination, depending on how certain you can be about the time the task is expected to start. Even a large internal trace table might wrap and overlay the data you want to see if you are not too sure about when the task should start.

Use the CETR transaction to set up the right tracing options. See Chapter 14, “Using traces in problem determination” on page 199 for guidance about setting up trace options.

Select special tracing for just your task, and disable tracing for all other tasks by setting the master system trace flag off. Set up special tracing for the level one trace points for the components that are likely to be used during the invocation of the task. The components you choose depend on how the task is initiated—by direct request from a terminal, or by automatic transaction initialization—but they should include loader domain (LD), program manager (PG), transaction manager (XM), and dispatcher domain (DS). Make sure that special tracing is disabled for all other components, to minimize the amount of trace data that is collected and the tracing overhead.

Now turn tracing on, and attempt to start your task. When you are sure that the time has passed when the output should have appeared, stop tracing, and format the trace data set.

If your transaction ran, you should see the following types of trace entries for your task and the programs associated with it:

1. Loader domain, when it loaded your program, if the program was not already in main storage.
2. Transaction manager, when it attached your task to the dispatcher.
3. Dispatcher domain, when your task got its first dispatch. You might also see subsequent entries showing your task being suspended, and then resumed.
4. Program manager, for any program management functions associated with your task.

If trace entries for any of these processes are missing, that should help you to find where the failure occurred.

Using EDF

If the transaction being tested requires a terminal, you can use EDF. You need two other terminals for input, as well as the one that the transaction requires (“tttt”). Use one of these others to put the transaction terminal under control of EDF, with:

```
CEDF tttt
```

At the remaining terminal, enter whatever transaction or sequence of transactions causes the one under test to be initiated. Wait long enough for it to start. If no

output appears at the second terminal, the transaction has not started. If you have not yet done so, consider using trace to get more information about the failure.

Using statistics

If no one else is using the transaction in question, you can tell from CICS statistics whether the program has been executed or not.

Use the command CEMT PERFORM STATISTICS RECORD (or your site replacement) before you test your transaction, using the TRANSACTION option:

```
CEMT PERFORM STATISTICS RECORD
[TRANSACTION]
```

This causes statistics on transactions that have been executed to be recorded in the DMF data set.

Now initiate the transaction and wait until it should have been executed. Repeat the CEMT PERFORM STATISTICS RECORD command, to get a new set of statistics written to the DMF data set. Format the data from the DMF data set. for the APPLID that interests you, and look at the statistics recorded before and after you attempted to execute the transaction. If the count for your transaction increased by 1, it was executed. If it remained the same, it was not executed.

Alternatively, if no one else is using the transaction, you can tell, using CEMT, whether the program is being executed. Use the command CEMT INQUIRE PROGRAM(xxxxxxxx) where xxxxxxxx is the program name. The screen presented to you includes a USECOUNT value. This value is the number of times that the program has been executed since the start of the current CICS session.

Now initiate the transaction and wait until it should have been executed. Repeat the CEMT INQUIRE PROGRAM(xxxxxxxx) and the USECOUNT value will have been incremented if the program has been executed.

Formatting the DMF data set. The statistics utility program, DFHSTUP, prepares and prints reports offline using the data recorded in the DMF data set. See the *CICS Operations and Utilities Guide* for information about using the DFHSTUP utility program.

When you format the DMF data set using DFHSTUP in order to look at the statistics relating to executed transactions and programs, you may find the following DFHSTUP control parameters useful:

- SELECT APPLID=
- COLLECTION TYPE=REQ
- TIME START= ,STOP=
- DATE START= ,STOP=

See the *CICS Operations and Utilities Guide* for details on how to code these parameters. If you correctly code these control parameters, you avoid the formatting of much information that might be unnecessary at this point.

Using CEBR

You can use CEBR to investigate your transaction if the transaction reads or writes to a transient data queue, or writes to a temporary storage queue. A change in such a queue is strong evidence that the transaction ran, provided that the environment is sufficiently controlled that nothing else could produce the same effect. You need to be sure that no other transaction that might be executed during your testing does the same thing.

The absence of such a change does not mean that the transaction did not run—it may have run incorrectly, so that the expected change was not made.

Using CECI

If your transaction writes to a file, you can use CECI before and after the transaction to look for evidence of the execution of your transaction. A change in the file means the transaction ran. If no change occurred, that does not necessarily mean that the transaction failed to run—it could have worked incorrectly, so that the changes you were expecting were not made.

Disabling the transaction

If your transaction requires a terminal, you can do the following. Use CEMT to disable the transaction under test, then do whatever causes the transaction to be initiated. You should get this message at the terminal where it is due to run:

DFHAC2008 *date time applid* **Transaction** *tranid* **has been disabled and cannot be used**

If you do not get this message, it is likely that your transaction did not start because of a problem with that terminal.

Investigating tasks initiated by ATI

In addition to the general techniques for all tasks described above, there are some additional ones for tasks that should have started by ATI.

Tasks to be started by ATI can be invoked in any of these ways:

- By issuing EXEC CICS START commands, even if no interval is specified
- By BMS ROUTE operations
- By writing to transient data queues with nonzero trigger levels.

There are many reasons why automatically initiated tasks could fail to start. Even when the CICS system is operating normally, an ATI transaction might fail to start for any of the following reasons:

- It might require a resource that is not available. The resource is usually a terminal, although it could be a queue.
- It might not be scheduled to start until some time in the future. START commands and output sent with BMS ROUTE are both subject to this sort of scheduling, but transactions started when transient data trigger levels are reached are not.

CICS maintains two chains for scheduling transactions that have been requested, but not started. They are the interval control element (ICE) chain, and the automatic initiate descriptor (AID) chain. The information contained in one or other of the chains can sometimes indicate why your task has failed to start.

The ICE chain

The ICE chain is used for tasks scheduled to start after some specified interval, for example on an EXEC CICS START command. You can locate it in the formatted system dump by looking at the ICP section. Look in field ICETRNID of each ICE (the 4-character transaction ID) to see if it relates to your task.

If you find an ICE for your task, look in field ICEXTOD. That will show you the expiration time of day. Does it contain the value you expect? If not, either the task which caused this one to be autoinitiated was in error, or there is a system problem.

The AID chain

The AID chain is used for tasks that are due to start immediately. Tasks are moved from the ICE chain to the AID chain as soon as the scheduled time expires, and they are placed there directly if there is no time delay requested. If a task needs a resource, usually a terminal, that is unavailable, the task remains on the AID chain until it can use the resource.

You can see the AIDs in the TCP section of the formatted system dump. Look in field AIDTRNID (the 4-character transaction ID) of each AID, to see if it relates to your task.

If you do find an AID that relates to your task, your task is scheduled to start, but cannot do so because the terminal is unavailable. Look in field AIDTRMID to find the symbolic ID of the terminal, and then investigate why the terminal is not available. One possibility is that the terminal is not in ATI status, because ATI(YES) has not been specified for it in the RDO TYPETERM resource definition.

Your transaction produced some output, but it was wrong

If your transaction produced no output at all, read “Your transaction produced no output at all” on page 164. For other types of wrong terminal output, read this section.

The origins of corrupted data

You get incorrect output to a terminal if data which is the object of the transaction becomes corrupted at some stage.

Figure 25 on page 170 illustrates how data flows between various CICS resources when a transaction is executed, and shows the points at which the data might become invalid.

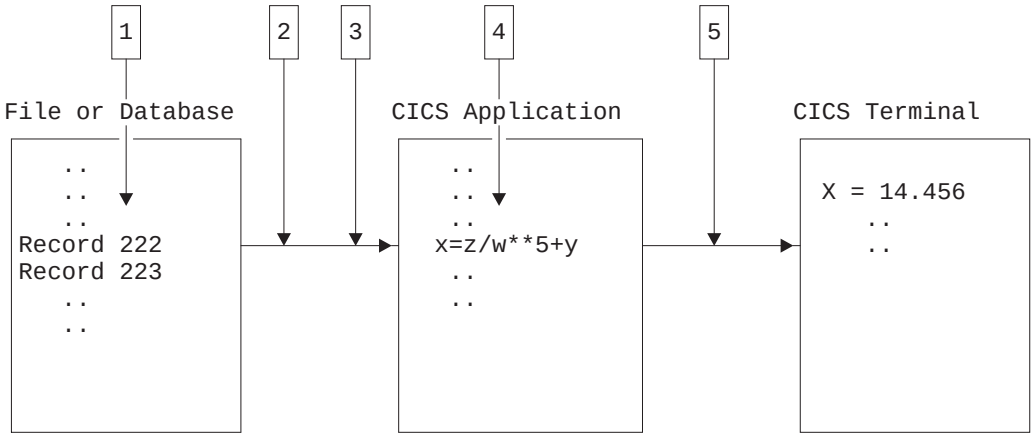


Figure 25. Where data might be corrupted in a transaction

The data might be corrupted at any of points 1 through 5, as it flows from file to terminal.

1. Data records might be incorrect, or they could be missing from the file.
2. Data from the file might be mapped into the program incorrectly.
3. Data input at the terminal might be mapped into the program incorrectly.
4. Bad programming logic might corrupt the data.
5. The data might be mapped incorrectly to the terminal.

Each of these possibilities will be dealt with in turn.

Were records in the file incorrect or missing?

You can check the contents of a file or database either by using CECI or by using a utility program to list off the records in question.

If you find bad data in the file or data set, the error is likely to have been caused by the program that last updated the records containing that data. If the records you expected to see are missing, make sure that your application can deal with a 'record not found' condition.

If the data in the file is valid, it must have been corrupted later on in the processing.

Was the data mapped correctly into the program?

When a program reads data from a file or a database, the data is put into a field described by a symbolic data declaration in the program.

Is the data contained in the record that is read compatible with the data declaration in the program?

Check each field in the data structure receiving the record, making sure in particular that the type of data in the record is the same as that in the declaration, and that the field receiving the record is the right length.

If the program receives input data from the terminal, make sure that the relevant data declarations are correct for that, too.

If there seems to be no error in the way in which the data is mapped from the file or terminal to the program storage areas, the next thing to check is the program logic.

Is the data being corrupted by bad programming logic?

To find out if data is being corrupted by bad programming logic in the application, determine the flow of data through the transaction. You can do this by using the following techniques:

- Desk checking your source code with the help of another programmer who is not familiar with the program. It is often possible for such a person to see weaknesses in the code which you have overlooked.
- Interactive tools. These allow you to look at the ways in which the data values being manipulated by your program change as the transaction proceeds.
  - CEDF is, perhaps, the most powerful interactive tool for checking your programming logic. You can use it to follow the internal flow from one CICS command-level statement to another. If necessary, you can add CICS statements such as EXEC CICS ASKTIME at critical points in your program, to see if certain paths are taken, and to check program storage values.
  - CECI allows you to simulate CICS command statements. Try to make your test environment match the environment in which the error occurred as closely as possible. If you do not, you might find that your program works with CECI, but not otherwise.
  - CEBR enables you to look at temporary storage and transient data queues, and to put data into them. This can be useful when many different programs use the queues to pass data.

    Note: When you use CEBR to look at a transient data queue, the records you retrieve are removed from the queue before they are displayed to you. This could alter the flow of control in the program you are testing. You can, however, use CEBR to copy transient data queues to and from temporary storage, as a way of preserving the queues if you need to.
- User tracing. This allows you to trace the flow of control and data through your program, and to record data values at specific points in the execution of the transaction. You could, for example, look at the values of counters, flags, and key variables during the execution of your program. You can include up to 4000 bytes of data on any trace entry, so this can be a powerful technique for finding where data values are being corrupted.

  For programming information about how you can invoke user tracing, see the description of the EXEC CICS ENTER TRACENUM command in the *CICS Application Programming Reference* manual.
- CSFE storage freeze. This can be used to freeze the storage associated with a terminal or a transaction so that it is not FREEMAINed at the end of processing. This can be a useful tool if, for example, you want to investigate possible storage violations. You need to get a transaction dump to look at the storage after you have run the task with storage freeze on.

  For long-running tasks, there is a possibility that a large amount of storage may be consumed because it cannot be FREEMAINed while storage freeze is on. For short-running tasks, however, there should be no significant overhead.

If, after using these techniques, you can find no fault with the logic of the program, the fault either lies with the way data is mapped to the terminal, or you could have missed some important evidence.

Is the data being mapped incorrectly to the terminal?

Incorrect data mapping to a terminal can have both application-related and system-related causes. If you are using BMS mapping, check the items below.

- Examine the symbolic map very carefully to make sure that it agrees with the map in the load module. Check the date and time stamps, and the size of the map.
- Make sure that the attributes of the fields are what they should be. For example:
  - An attribute of DARK on a field can prevent the data in the field from being displayed on the screen.
  - Failing to turn on the modified data tag (MDT) in a field might prevent that field from being transmitted when the screen is read in.

    Note: The MDT is turned on automatically if the operator types data in the field. If, however, the operator does not type data there, the application must turn the tag on explicitly if the field is to be read in.
- If your program changes a field attribute byte, or a write control character, look at each bit and check that its value is correct by looking in the appropriate reference manual for the terminal.

Chapter 10. Dealing with storage violations

The first section in this chapter describes the CICS facilities for preventing storage violations. The rest of the chapter describes problem determination techniques for storage violations once they occur.

Avoiding storage violations

CICS provides two facilities that help prevent storage violations.

CICS subsystem storage protection
: prevents user application programs from directly overwriting CICS code and control blocks.

Command protection
: prevents CICS, when processing an EXEC CICS command, from overwriting storage that the issuing transaction could not itself directly overwrite.

Even if your system uses all the CICS storage protection facilities, CICS storage violations can occur in certain circumstances in systems using storage protection. For example:

- An application program could contain the necessary instructions to switch to CICS key and modify CICS storage.
- An application program could contain the necessary instructions to switch to the basespace and modify the storage of other transactions.
- An application program could be defined with EXECKEY(CICS) and could thus modify CICS storage and the storage of other transactions.
- An application could overwrite one or more storage check zones in its own task-lifetime storage.

To gain the full benefit of CICS storage protection, you need to examine the storage needs of individual application programs and control the storage key definitions that are used.

When CICS detects and prevents an attempted storage violation, the name of the abending program and the address of the area it tried to overwrite are passed to the program error program (DFHPEP). For programming information about DFHPEP, see the *CICS Customization Guide*.

If a storage violation does occur in your system, read the rest of this chapter.

© Copyright IBM Corp. 1979, 1999

Two kinds of storage violation

Storage violations can be divided into two classes, namely those detected and reported by CICS, and those not detected by CICS. They require different problem determination techniques.

CICS-detected violations are identified by the following message sent to the console:

DFHSM0102 *applid* **A storage violation (code X'***code***') has been detected by module** *modname*.

If you have received this message, turn first to the description of message DFHSM0102 in the *VSE/ESA Messages and Codes Volume 3* manual to see an explanation of the value of '*code*', telling you how CICS detected the storage violation. Then return to this section, and read "CICS has detected a storage violation."

Storage violations not detected by CICS are less easy to identify. They can cause almost any sort of symptom. Typically, you may have got a program check with a condition code indicating 'operation exception' or 'data exception', because the program or its data has been overlaid. Otherwise, you might have obtained a message from the dump formatting program saying that it had found a corrupted data area. Whatever the evidence for the storage violation, if it has not been detected by CICS, turn to "Storage violations that affect innocent transactions" on page 181.

CICS has detected a storage violation

CICS can detect storage violations when:

1. The duplicate storage accounting area (SAA) or the initial SAA of a TIOA storage element has become corrupted.
2. The leading storage check zone or the trailing storage check zone of a user-task storage element has become corrupted.

CICS detects storage violations involving TIOAs by checking the SAA chains when it receives a command to FREEMAIN an individual element of TIOA storage, at least as far as the target element. It also checks the chains when it FREEMAINs the storage belonging to a TCTTE after the last output has taken place. CICS detects storage violations involving user-task storage by checking the storage check zones of an element of user-task storage when it receives a command to FREEMAIN that element of storage. It also checks the chains when it FREEMAINs all the storage belonging to a task when the task ends.

The storage violation is detected *not at the time it occurs*, but *only when the SAA chain or the storage check zones are checked*. This is illustrated in Figure 26 on page 176, which shows the sequence of events when CICS detects a violation of a user task storage element. The sequence is the same when CICS detects a violation of a TIOA storage element.

The fact that the SAA or storage check zone is overlaid some time before it is detected does not matter too much for *user* storage where the trailing storage

check zone has been overlaid, because the transaction whose storage has been violated is also very likely to be the one responsible for the violation. It is fairly common for transactions to write data beyond the end of the allocated area in a storage element and into the check zone. This is the cause of the violation in Figure 26 on page 176.

The situation could be more serious if the leading check zone has been overlaid, because in that case it could be that some other unrelated transaction was to blame. However, storage elements belonging to individual tasks are likely to be more or less contiguous, and overwrites could extend beyond the end of one element and into the next.

If the leading storage check zone was only overwritten by chance by some other task, the problem might not be reproducible. On other occasions, other parts of storage might be affected. If you have this sort of problem, you need to investigate it as though CICS had not detected it, using the techniques of “Storage violations that affect innocent transactions” on page 181.

Finding the offending transaction when the duplicate SAA of a TIOA storage element has been overlaid might not be so straightforward. This is because TIOAs tend to have much longer lifetimes than tasks, because they wait on the response of terminal operators. By the time the storage violation is detected, the transaction that caused it is unlikely to still be in the system. However, the techniques for CICS-detected violations still apply.

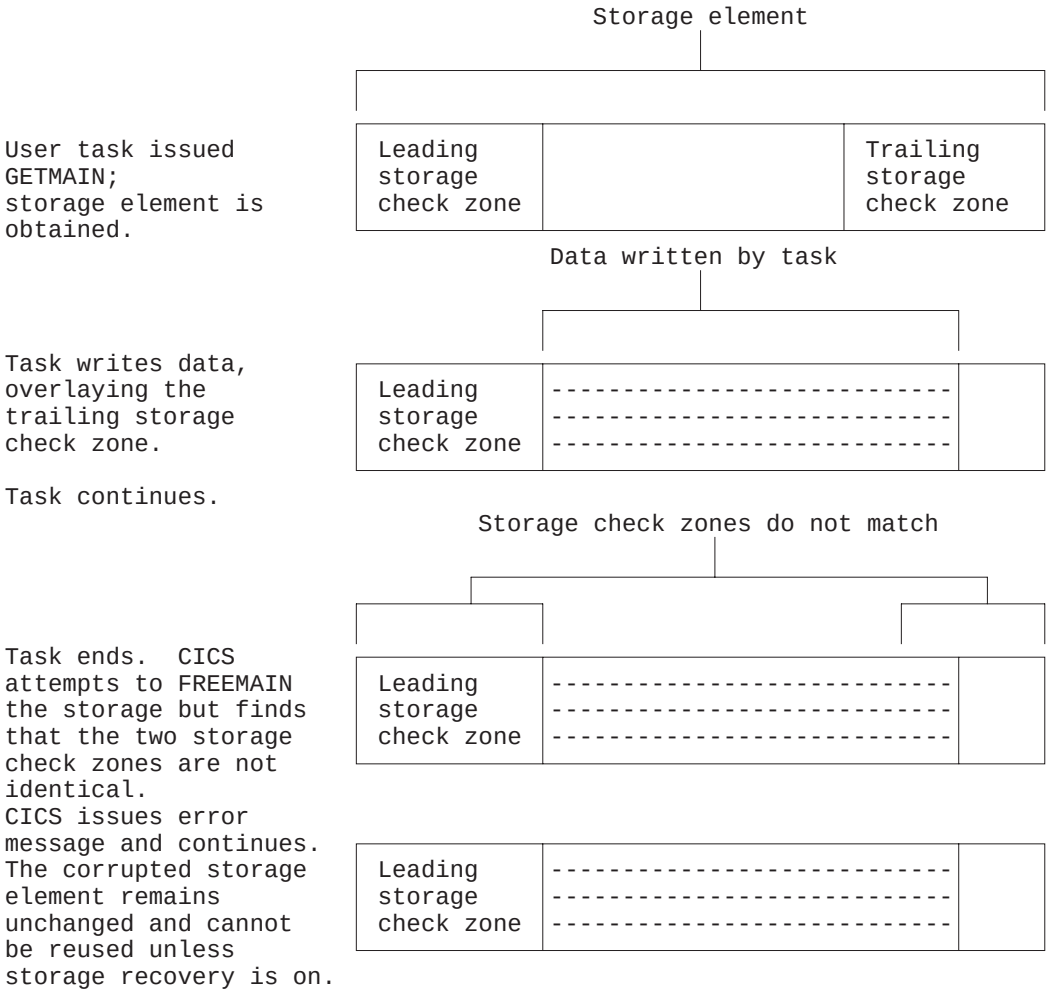


Figure 26. How user-storage violations are committed and detected

Note: For storage elements with SAAs, the address that is returned on the GETMAIN request is that of the leading SAA; for storage elements with storage check zones, the address that is returned is that of the beginning of usable storage.

What happens when CICS detects a storage violation

When CICS detects a storage violation, it makes an exception trace entry in the internal trace table, issues message DFHSM0102 and takes a CICS system dump, unless you have suppressed dumping for system dump code SM0102. If you have suppressed dumping for this dump code, re-enable it and attempt to reproduce the error. The system dump is an important source of information for investigating CICS-detected storage violations.

If storage recovery is on (STGRCVY=YES system initialization parameter), the corrupted SAAs or check zones are repaired and the transaction continues. See “Storage recovery” on page 183.

If storage recovery is not on, CICS abends the transaction whose storage has been violated (if it is still running). If the transaction is running when the error is detected and if dumping is enabled for the dump code, a transaction dump is taken.

If you received a transaction abend message, read “What the transaction abend message can tell you.” Otherwise, go on to “What the CICS system dump can tell you.”

What the transaction abend message can tell you

If you get a transaction abend message, it is very likely that CICS detected the storage violation when it was attempting to satisfy a FREEMAIN request for user storage. Make a note of the information the message contains, including:

- The transaction abend code
- The identity of the transaction whose storage has been violated
- The identity of the program running at the time the violation was detected
- The identity of the terminal at which the task was started

Because CICS does not detect the overlay at the time it occurs, the program identified in the abend message is probably not the one in error. However, it is likely that it issued the FREEMAIN request on which the error was detected. One of the other programs in the abended transaction might have violated the storage in the first place.

What the CICS system dump can tell you

Before looking at the system dump, you must format it using the appropriate formatting keywords. The ones you need for investigating storage violations are:

- TR, to get you the internal trace table
- TCP, to get you terminal-related areas
- AP, to get you the TCAs and user storage

The dump formatting program reports the damaged storage check zone or SAA chain when it attempts to format the storage areas, and this can help you with diagnosis by identifying the TCA or TCTTE owning the storage.

When you have formatted the dump, take a look at the data overlaying the SAA or storage check zone to see if its nature suggests which program put it there. There are two places you can see this, one being the exception trace entry in the internal trace table, and the other being the violated area of storage itself.

Look first at the exception trace entry in the internal trace table to check that it shows the data overlaying the SAA or storage check zone. Does the data suggest what program put it there? Remember that the program is likely to be part of the violated transaction in the case of user storage. For terminal storage, you probably have more than one transaction to consider.

As the SAAs and storage check zones are only 8 bytes long, there might not be enough data for you to identify the program. In this case, find the overlaid data in the formatted dump. The area is pointed to in the diagnostic message from the dump formatting program. The data should tell you what program put it there, and, more importantly, what part of the program was being executed when the overlay occurred.

If the investigations you have done so far have enabled you to find the cause of the overlay, you should be able to fix the problem.

What to do if you cannot find what is overlaying the SAA

The technique described in this section enables you to locate the code responsible for the error by narrowing your search to the sequence of instructions executing between the last two successive old-style trace entries in the trace table.

You do this by forcing CICS to check the SAA chain of terminal storage and the storage check zones of user-task storage every time an old-style trace entry is made from AP domain. These types of trace entry have point IDs of the form AP 00xx, “xx” being two hexadecimal digits. Storage chain checking is not done for new-style trace entries from AP domain or any other domain. (For a discussion of old and new-style trace entries, see Chapter 14, “Using traces in problem determination” on page 199.)

The procedure has a significant processing overhead, because it involves a large amount of tracing. You are likely to use it only when you have had no success with other methods.

How you can force storage chain checking

You can force storage chain checking either by using the CSFE DEBUG transaction, or by using the CHKSTSK or CHKSTRM system initialization parameters. See the *CICS System Definition Guide* for more information about these parameters. Tracing must also be active, or CICS will do no extra checking. The CSFE transaction has the advantage that you need not bring CICS down before you can use it.

Table 28 on page 179 shows the CSFE DEBUG options and their effects. Table 29 shows the startup overrides that have the same effects.

Table 28. Effects of the CSFE DEBUG transaction	
CSFE syntax	**Effect**
CSFE DEBUG, CHKSTSK=ALL	This checks storage check zones of all storage areas on the transaction storage chain for every task. You need to use CSFE DEBUG,CHKSTSK=ALL if one task's storage check zone is being overlaid by another task.
CSFE DEBUG, CHKSTSK=CURRENT	This checks storage check zones for all storage areas on the transaction storage chain for the current task only. If a task is overlaying one of the storage check zones of its own user storage, use CSFE DEBUG,CHKSTSK=CURRENT
CSFE DEBUG, CHKSTRM=CURRENT	This checks SAAs for all TIOAs linked off the current TCTTE. Use this if the SAA of a TIOA has been overlaid.
CSFE DEBUG, CHKSTSK=NONE	This turns off storage zone checking for transaction storage areas.
CSFE DEBUG, CHKSTRM=NONE	This turns off SAA checking for TIOAs.

Table 29. Effects of the CHKSTSK and CHKSTRM system initialization parameters	
Override	**Effect**
CHKSTSK=ALL	As CSFE DEBUG,CHKSTSK=ALL
CHKSTSK=CURRENT	As CSFE DEBUG,CHKSTSK=CURRENT
CHKSTRM=CURRENT	As CSFE DEBUG,CHKSTRM=CURRENT
CHKSTSK=NONE	As CSFE DEBUG,CHKSTSK=NONE. This override is the default.
CHKSTRM=NONE	As CSFE DEBUG,CHKSTRM=NONE. This override is the default.

Your strategy should be to have the minimum tracing that will capture the storage violation, to reduce the processing overhead and to give you less trace data to process. Even so, you are likely to get a large volume of trace data, so direct the tracing to the auxiliary trace data sets. For general guidance about using tracing in CICS problem determination, see Chapter 14, "Using traces in problem determination" on page 199.

You need to have only level-1 tracing selected, because no user code is executed between level-2 trace points. However, you do not know which calls to CICS components come before and after the offending code, so you need to trace all CICS components in AP domain. (These are the ones for which the trace point IDs have a domain index of "AP".) Set level-1 tracing to be special for all such

components, so that you get every AP level-1 trace point traced using special task tracing.

If the trailing storage check zone of a user-storage element has been overlaid, select special tracing for the corresponding transaction only. This is because it is very likely to be the one that has caused the overlay.

If the duplicate SAA of a TIOA has been overlaid, you need to select special tracing for all tasks associated with the corresponding terminal, because you are not sure which has overlaid the SAA. It is sufficient to select special tracing for the terminal and standard tracing for every transaction that runs there, because you get special task tracing with that combination. (See Figure 35 on page 212.)

Your choice of terminal tracing depends on where the transaction is likely to be initiated from. If it is only ever started from one terminal, select special tracing for that terminal alone. Otherwise, you need to select special tracing for every such terminal.

When you have set up the tracing options and started auxiliary tracing, you need to wait until the storage violation occurs.

What happens after CICS detects the storage violation?

When the storage violation is detected by the storage violation trap, storage checking is turned off, and an exception trace entry is made. If dumping has not been disabled, a CICS system dump is taken. The following message is sent to the console:

DFHSM0103 *applid* **STORAGE VIOLATION (CODE X'***code***') HAS BEEN DETECTED BY THE STORAGE VIOLATION TRAP. TRAP IS NOW INACTIVE**.

The value of '*code*' is equal to the exception trace point ID, and it identifies the type of storage that was being checked when the error was detected.

Perform the following steps:

1. Format the system dump using the formatting keyword TR, to get the internal trace table.
2. Locate the exception trace entry made when the storage violation was detected, near the end of the table.
3. Scan back through the table, and find the last old-style trace entry (AP 00xx). The code causing the storage violation was being executed between the time that the trace entry was made and the time that the exception trace entry was made.
4. If you have used the CHKSTSK=CURRENT option, can locate the occurrence of the storage violation only with reference to the last old-style trace entry for the current task. For greater accuracy in locating the occurrence of the violation, you might need to use the CHKSTSK=ALL option. However, keep in mind the additional overhead of running with CHKSTSK=ALL, and use it only if you need it.
5. Identify the section of code that was being executed between the two trace entries from the nature of the trace calls. You then need to study the logic of the code to find out how it caused the storage violation.

For suggestions on programming errors that might have caused your particular problem, look at the list of common ones given in “Programming errors that can cause storage violations” on page 182.

Storage violations that affect innocent transactions

Storage violations that affect innocent transactions—that is, transactions that do not cause the violation—usually go undetected by CICS. However, occasionally CICS detects that the initial SAA of a TIOA element or the storage check zone of a user-storage element has been overlaid by a task that does not own it.

If they are reproducible, storage violations of this type typically occur at specific offsets within structures. For example, the start of an overlay might always be at offset 30 from the start of a field.

The most likely cause of such a violation is a transaction writing data to a part of the DSAs that it does not own, or possibly FREEMAINing such an area. The transaction might previously have GETMAINed the area and then FREEMAINed it before writing the data, or addressability might otherwise not have been correctly maintained by an application. Another possible reason is that an ECB might have been posted by a transaction after the task that was waiting on it had been canceled.

Storage violations affecting innocent transactions are, in general, more difficult to resolve than those that are detected by CICS. Often, you become aware of them long after they occur, and then you need a long history of system activity to find out what caused them.

A strategy for storage violations affecting innocent transactions

In this case, the storage violation has been caused by a program writing to an area it does not own, but you probably have no idea at the outset which program is at fault. Look carefully at the content of the overlay before you do any other investigation, because it can help you to identify the transaction, program, or routine that caused the error. If it does not provide the information you need, use CICS tracing to collect a history of all the activities that reference the affected area.

The trace table must go back as far as task attach of the program causing the overlay, because that trace entry relates the transaction’s identity to the unit of work number used on subsequent entries— this could require a very large trace table. Internal trace is not suitable, because it wraps when it is full and it then overwrites important trace entries.

You should use auxiliary trace for recording long periods of system activity, as you can specify very large auxiliary trace data sets which do not wrap when full.

If you have no idea which transaction is causing the overlay, you need to trace the activities of every transaction. This impacts performance, because of the processing overhead.

A procedure for resolving storage violations affecting innocent transactions

1. Ensure that level-1 trace points are in the special set for all CICS components. Select special tracing for all user tasks, by setting up standard tracing for all user transactions and all terminals, and disable standard tracing by setting the master system trace flag off.
2. Use the CETR transaction to set up the tracing options, and select auxiliary trace as the trace destination.
3. When you get the symptoms that tell you that the storage violation has occurred, take a system dump—unless the error causes a system dump to be taken.
4. Format the system dump, and format and print the auxiliary trace data set. If you know which area of storage the violation occurred in, you can use appropriate dump formatting keywords. Otherwise, you need to format the entire system dump. The dump formatting program may report that it has found some incorrect data. If not, you need to find the overlaid area by other means.
5. Locate all the entries in the trace table that address the overlaid area. Operations involving GETMAIN and FREEMAIN in particular are likely pointers to the cause of the error.
6. When you have found a likely trace entry, possibly showing a GETMAIN or FREEMAIN addressing the area, find the ID of the associated transaction by locating the trace entry for TASK ATTACH. Rather than locating this manually, it is probably better to reformat the auxiliary trace data set selectively to show just trace entries corresponding to the task's unit of work.
7. Take a look at all the programs belonging to the transaction. It is likely that one of these caused the overlay, and you need to consider the logic of each to see if it could have caused the error. This is a long job, but it is one of the few ways of resolving a storage violation affecting an innocent transaction.

What to do if you still cannot find the cause of the overlay

If you are unable to identify the cause of the storage violation after carrying out the procedures of the preceding section, contact your IBM Support Center. They may suggest coding a global trap/trace exit to detect the storage violation.

Programming errors that can cause storage violations

The purpose of this section is to outline a number of commonly occurring programming errors that can cause storage violations.

1. Failing to GETMAIN sufficient storage. This is often caused by failure to recompile all the programs for a transaction after a common storage area has been redefined with a changed length.
2. Runaway subscript. Make sure that your tables can only grow to a finite size.
3. Writing data to an area after it has been FREEMAINed.

   When a task FREEMAINs an area that it has been addressing, it can no longer write data to the area without the risk of overwriting some other data that might subsequently be there.

4. Hand posting an ECB for a canceled task.

   If a task waiting on a CICS ECB is canceled, and then a transaction attempts to hand post the ECB when the resource being waited on becomes available, it may corrupt data belonging to some unrelated activity if the area once occupied by the ECB has been reused.

Storage recovery

The STGRCVY system initialization parameter enables you to vary the action taken by CICS on detection of a storage violation.

In normal operation, CICS sets up four task-lifetime storage subpools for each task. Each element in the subpool starts and ends with a check zone that includes the subpool name. At each FREEMAIN, and at end of task, CICS inspects the check zones and abends the task if either has been overwritten.

Terminal input-output areas (TIOAs) have similar check zones, each of which is set up with the same value. At each FREEMAIN of a TIOA, CICS inspects the check zones and abends the task if they are not identical.

If CICS is initialized with STGRCVY(YES), the overwriting of check zones is treated differently. After the system dump has been taken for the storage violation, CICS resets the check zones to their initial value and the task continues execution.

STGRCVY(NO) is the default.

Chapter 11. Dealing with XRF errors

If you have a problem when using XRF, it might not be easy to determine whether the active or the alternate system is at fault. In diagnosing these problems, the message content and sequence can be very useful. The CICS availability manager data sets hold these messages (see “The CAVM data sets” on page 190).

Symptoms of problems in an XRF complex

This section gives examples of the symptoms that can occur in your CICS system when using the extended recovery facility (XRF). For each symptom, a list of questions leads to the possible causes, indicating the nature of the error. If the error is a CICS error rather than a user error, then the module in error is shown.

The symptoms are divided as follows:

- “Active system does not initialize”
- “Alternate system does not initialize”
- “Takeover starts, but does not complete” on page 186
- “Not all terminals recovered after a takeover” on page 187
- “Unexpected takeover” on page 187
- “Takeover does not occur when expected” on page 187
- “Related systems are not taken over” on page 188
- “Alternate system terminates unexpectedly” on page 188
- “Overseer does not restart a job that had failed” on page 189
- “Unable to log on to CICS” on page 189

Active system does not initialize

1. Does VSE indicate that the active system is waiting for data sets?

   Check that the necessary data sets have been correctly specified.

2. Have the CAVM data sets (the control and message data sets) been set up correctly? For more information about the data sets, see “The CAVM data sets” on page 190.

3. For a multi-VSE environment, have the POWER shared spool requirements been met?

4. Has the active system signed on to the CAVM?

5. If not, is another job signed on to the CAVM as an active system, or is takeover in progress?

6. Is the active system signed on to the CAVM, and an alternate system in the process of taking over and awaiting a response to a message?

Alternate system does not initialize

1. Does VSE indicate that the alternate system is waiting for data sets?

   Check that the necessary data sets have been correctly specified.

2. Have the CAVM data sets (the control and message data sets) been set up correctly? For more information about the data sets, see “The CAVM data sets” on page 190.

© Copyright IBM Corp. 1979, 1999

3. For a multi-VSE environment, have the POWER shared spool requirements been met?
4. Is an alternate system already signed on to the CAVM?
5. Have you started VTAM?

   Unlike the active system, the alternate system cannot initialize without VTAM.

Takeover starts, but does not complete

1. Has the takeover been accepted by the alternate system?
2. Is the alternate system waiting for confirmation that the active system has ended (particularly in a CPC failure case)?

   Even though the alternate system has successfully issued the CANCEL, the active system may be unable to terminate abnormally, for example it may be necessary for the system operator to issue the FORCE command.

   If the active system has ended, the alternate system may be unable to determine from POWER that it has. If the CPC has failed, POWER may still indicate that the active job is running.

   In these cases, a reply is probably outstanding to message DFHXA6561D on the system operator console, or to message DFHXA6577D or DFHXA6578D if an error has been detected with the CLT contents. See the *VSE/ESA Messages and Codes Volume 3* manual for guidance information about the responses to these messages.
3. Did the alternate system attempt to cancel the active system?

   When the alternate system issues the CANCEL command, this should be displayed on both system consoles in a two-processor configuration. If VSE on the alternate system processor did not accept the command, message DFHXA6560I is displayed by the alternate system.

   If there is a failure of POWER communication between the active and alternate processors, the CANCEL command may not have been routed.

   If there are errors in the CLT, the alternate system may not be able to construct the appropriate CANCEL command.

   In each of these cases, a reply to messages DFHXA6561D, DFHXA6577D or DFHXA6578D may be required. See the *VSE/ESA Messages and Codes Volume 3* manual for guidance information about the responses to these messages.
4. Do you have the correct definitions in the CLT?

   If CLT errors were detected, the alternate system may not be able to construct the CANCEL command. A previous message should be displayed on the alternate system processor to report the error with the CLT contents.

   A reply to messages DFHXA6561D, DFHXA6577D or DFHXA6578D may be required. See the *VSE/ESA Messages and Codes Volume 3* manual for guidance information about the responses to these messages.

Not all terminals recovered after a takeover

1. In a multi-VSE environment, do the control units need switching?
2. Are the necessary VTAM cross-domain definitions active and available?
3. Should the reconnect transaction (CXRE) have run yet?

   If CXRE could not be scheduled, messages DFHXG6485, DFHXG6487, or DFHXG6488 are issued. If CXRE was delayed, to allow physical switching of terminals to be completed, message DFHXG6481I is issued, telling you how long the delay will be. This delay includes the delay specified in the SIT parameter AUTCONN.
4. Did the reconnect transaction (CXRE) start up on time?

   Message DFHXG6490I is issued at the start of every reconnection pass. When CXRE has reconnected all the terminals that it believes it should reconnect, it ends the pass with message DFHXG6484I. Otherwise, if it is unable to reschedule itself, it issues the message DFHXG6489.
5. Is there a combination of long-running tasks updating recoverable resources without syncpointing, and a high value in the AKPFREQ system initialization parameter?

   With this combination, a terminal or session that is installed, subsequently reinstalled, and then acquired, might not be reacquired after a takeover. If this happens, you should ensure that long-running tasks take regular syncpoints, and you should set a lower AKPFREQ value.

Unexpected takeover

1. Did someone stop the active system's processor, with TAKEOVER=AUTO coded in the SIT for the alternate system?
2. Is the ADI value specified for the alternate system too low? For guidance about the ADI system initialization parameter, see the *CICS System Definition Guide*. For general details of the system initialization parameters you need to define for XRF, see the *CICS XRF Guide*.
3. In a two-processor system, has there been some presumably unexpected processing that stopped the active system from running?

   Note: If such processing is expected, you can suspend surveillance by using the command CEBT SET SURVEILLANCE OFF, or change the alternate system to TAKEOVER=COMMAND by using CEBT SET TAKEOVER COMMAND for the duration of the activity.
4. Has a badly coded CLT initiated unnecessary takeovers?

Takeover does not occur when expected

1. Was the correct TAKEOVER value in effect for the alternate system?
2. If the CEBT PERFORM TAKEOVER command was entered, was it accepted?
3. If you are expecting an automatic takeover, is the ADI value set too high?
4. If you have set up for an automatic takeover, was the missing surveillance signal detected by the alternate system?
5. Did the active system sign off abnormally?

   If so, did the alternate system detect this?

6. Did the alternate system have a large backlog of messages in the CAVM message file?

   If the backlog becomes too large, the alternate system becomes invalid. Do you need a larger message file?

7. Is the alternate system clock significantly behind the active system clock? This can cause the alternate system to wait before completing the takeover. See messages DFHXG6682I and DFHXG6683I.

8. Is a CEBT SET SURVEILLANCE OFF command still in force, if one has been previously issued? Is the alternate system in TAKEOVER=COMMAND status set by a previously issued CEBT SET TAKEOVER COMMAND command?

Related systems are not taken over

1. Has the master/coordinator alternate system run its CLT successfully?

   If errors were detected in the CLT, message DFHXA6576I should have been issued by the alternate system. See the *VSE/ESA Messages and Codes Volume 3* manual for an explanation of this message.

2. Do you have the correct definitions in the CLT, so that the other alternate systems are instructed to take over?

   If other alternate systems of related systems should have initiated a takeover as a result of a command entry in the CLT, but did not, the appropriate command entries may not be correctly specified in the CLT, or the set of commands may be incomplete.

   System operator commands issued as part of CLT processing are displayed on the system console of the alternate system processor. If an error was detected in a command, there should be an associated VSE error message displayed on this console. System operator commands in the CLT are not interpreted by CICS.

   The CLT contents may not be correct for use by this alternate system. For example, the appropriate APPLID may not be correctly specified in the CLT, in which case message DFHXA6567I is displayed on the system console reporting the error.

Alternate system terminates unexpectedly

1. Did the active system shut down normally, thus causing the alternate system to shut down normally?

2. Was the alternate system shutdown caused by a CAVM event?

   This could be because a new active system has been initialized, thus invalidating the existing alternate system.

3. Did the alternate system's VTAM fail, thus causing the alternate system to terminate?

   The alternate system does not complete initialization until VTAM is available and the VTAM ACB has been opened.

   Similarly, if the VTAM in the alternate system fails while the alternate system is tracking the active system, the alternate system cannot continue, and it is abended. Bring up VTAM again, and restart the alternate system.

4. Did the alternate system become invalid because of an unacceptable message backlog in the CAVM message file?

Overseer does not restart a job that had failed

1. Is 'restart in place' disabled?
2. Is 'restart in place' not active for this system?
3. Is the job to be restarted not defined to the overseer?

   If the overseer determines that a takeover is taking place, or is likely to take place, it does not perform a restart in place.
4. Was the overseer unable to open/read the CAVM data sets associated with this job?

Unable to log on to CICS

1. Is the VTAM USERVAR set to the correct value?

   The VTAM USERVAR is the CICS generic APPLID. You can use the VTAM commands D NET,USERVAR and F NET,USERVAR to inspect, and change, the setting of the USERVAR whose ID is the APPLID used in the failing logon attempts.
2. Has CICS issued the necessary F NET,USERVAR command?

   The F NET,USERVAR command is logged by both VSE and VTAM. CICS issues message DFHXG6479 if it is unable to issue the F NET,USERVAR command.
3. Has the new USERVAR value been propagated around the network after initialization of the active system, or after a takeover?
4. Are the necessary VTAM cross-domain definitions not active, or not available?
5. Did you set up the specific and generic APPLIDs correctly in your system initialization parameters? Are they the right way round?
6. Are you trying to log on to the generic? Use the generic APPLID during logon.

Debugging the overseer sample program

If the overseer detects an error, it may put out a diagnostic message, or take a snap dump, or do both of these things. You can request a snap dump of the overseer's partition using the CEMT PERFORM SNAP command.

The documentation for messages put out by the overseer is in the prolog of the source for the sample program, DFH$AXRO.

Return and reason codes for some types of failure are saved in the RADBS control block, and within individual GENDS control blocks if the error is connected with a particular XRF pair of CICS systems.

Notes:

1. Some overseer problems are asynchronous, and post a user ECB when they complete. Be careful not to reuse user areas before the events complete.
2. There is one system key area, and one user key area, managed by the overseer authorized services. You can find the assembler language DSECTs and a description of the logic used in module DFHWOSB.

XRF trace

For information about XRF trace, see "CICS XRF tracing" on page 221.

The CAVM data sets

The control and message data sets, used by the CICS availability manager (CAVM), are both VSAM ESDS data sets which are accessed using control interval processing with user buffering. The control intervals (CIs) have the standard VSAM format with CI definition fields (CIDFs) and record definition fields (RDFs). All data in a given control or message data set relates to a single generic APPLID.

The **control data set** contains a CI for recording CAVM's view of the overall state of the generic APPLID (the state management record), and a status CI for each potential member system (the active and alternate systems).

The **message data set** contains secondary status CIs, to be used when a system is unable to write its status CI in the control data set, and enough message CIs to hold the maximum anticipated backlog of messages from the active system to the alternate system.

At most times, when the active and alternate systems are running normally, serialization of access to the data sets is unnecessary, because the state management record remains unchanged, and all other CIs in the two data sets are owned (updated) by only one of the member systems. However, when the state changes, at SIGNON, TAKEOVER or SIGNOFF, the state management record is changed, and ownership of other CIs may also change. VSE LOCK is used to serialize updates of the state management record, and any changes of ownership of other CIs are effected under this LOCK. VSE UNLOCK is used to free the data set, so that any system can update it again.

The contents of the data sets are described in more detail in the following sections of this chapter and may be useful when the problem cannot be easily ascribed to the active system or the alternate system. For example, a terminal might be lost (not switched), or the system waiting indefinitely.

Contents of the control data set

The control data set contains the following sequence of CIs:

1. Control CI
2. State management record CI
3. Primary active system status CI
4. Primary alternate system status CI.

The control CI is used for validation. It contains:

```
Bytes      Contents
0- 4       CAVM data set format number
8-14       Filename (DFHXCTL for control data set,
                     DFHXMSG for message data set)
15-22      Generic APPLID
23-42      Unique identification comprising:
            23-30    TOD clock at time of formatting by CAVM
            31-33    PWR
            34       POWER SYSID of formatting system
            35-42    IPL TOD
```

A control CI with the same format is also present in the message data set. During the CAVM SIGNON procedure, CAVM writes both CIs as part of its formatting operation if both data sets were previously unused. Otherwise, it validates the two CIs by checking that:

- The CAVM code level and data-set format are compatible.
- They have the same unique identification data.
- The filename value is appropriate for the intended use of the data set.
- The generic APPLID is the same as that of the system attempting to sign on.

The state management record contains information about the overall state of the generic APPLID. It consists of a fixed part, followed by a number of job description parts, one for each potential member system. The format of the fixed part is defined by the DSECT SMDESCR in DFHWSMDS. The format of a job description part is defined by the DSECT WSJDESC in DFHWSMDS.

The status CIs contain information about a particular member system. Unlike the state management record, they are continually updated. The status data may be recorded in a corresponding CI in the message data set if a system is for some reason unable to update its control data set status CI. The preferred place for status information is the control data set.

Each status CI consists of a fixed part which describes that system, and may be of interest to any partner system. This is followed by parts containing information directed to a specific partner system. Since the implementation allows only one alternate system, there is in fact only one instance of these specific parts in each CI.

The fixed part of the status CI consists of a 12-byte status record header, followed by data described by the DSECT WSAS in DFHWSADS. The format of the status record header is:

```
Bytes        Contents
0- 7         TOD clock value at time of issuing write request
8-11         Write sequence number
```

The sequence numbers of the status CIs in the control data set are independent. However corresponding status CIs in the control and message data sets share a sequence number. Except for the special case when both contain their initial value (zero), corresponding status CIs should always contain different sequence numbers with the higher value in the CI containing the more up-to-date data.

The following part of the status CI contains in order:

1. Data described by DSECT WSAR in DFHWSADS.
2. Either invalidation data for the alternate system (DSECT WSARIV in DFHWSADS) if it is an active system status CI, or a takeover message for the active system (DSECT WSARTM in DFHWSADS) if it is an alternate system status CI.
3. Expedited message data from PUTREQ and PUTRSP requests (DSECT WSARQR in DFHWSADS).

Contents of the message data set

The message data set contains the following sequence of CIs:

1. Control CI
2. Unused CI (set to zeros)
3. Secondary active system status CI
4. Secondary alternate system status CI
5. Message data CIs up to the end of the space allocated.

The control CI and status CIs have already been described in the section on the control data set.

The message data CIs form a wraparound store of message records generated as the result of PUTMSG requests issued by the active system to the CAVM message manager. The active system's write cursor and the alternate system's initial-read cursor are both located in the active system's status CI (see the DSECTs WSAR and WSAS in DFHWSADS.) The alternate system's current-read cursor is located in the alternate system's status CI (see the DSECT WSAR in DFHWSADS.)

The CIs are in standard VSAM format, and each message is represented by a single nonspanned ESDS record. CIDF and RDF control fields are maintained accordingly. The message manager does not attempt to combine RDFs for adjacent records of equal length, consequently there is one RDF for each record in the CI.

Each message CI contains:

1. A control record
2. Zero or more message records.

Chapter 12. External CICS interface

The CICS external CICS interface produces messages with a format of DFHEXnnnn.

For full details of all CICS messages, see the *VSE/ESA Messages and Codes Volume 3*.

The external CICS interface outputs trace to the internal trace table, which is located in the non-CICS VSE batch partition. Trace data is formatted and included in any dumps produced by the external CICS interface.

Trace entries issued by the external CICS interface are listed in the *CICS Trace Entries* manual.

The external CICS interface produces VSE SDUMPS for some error conditions. These dumps contain all the external CICS interface control blocks, as well as trace entries.

A user-replaceable program, DFHXCTRA, is available for use under the guidance of IBM service personnel. It is the equivalent of DFHTRAP used in CICS. It is invoked every time the external CICS interface writes a trace entry. The actions of the CICS-supplied DFHXCTRA are, on a pipe FREEMAIN error, to:

1. Make a trace entry
2. Take an SDUMP
3. Skip writing the current trace entry
4. Disable itself.

© Copyright IBM Corp. 1979, 1999

Chapter 13. Dealing with MRO problems

This chapter discusses problems with multiregion operation (MRO).

If an error is suspected in communication between System A and System B, the problem can be determined by looking at either system. The following procedure is applied here to System A, but could equally well be applied to System B:

1. First, look at the field CSACRBA in the CSA optional features list in System A. If CSACRBA is zero, the interregion communication routine is not active in that system.
2. If CSACRBA is not zero, examine the TCTSEs in System A and find the TCTSE for System B. In this TCTSE, TCSEIRCF is the flag byte that indicates the state of communication between the two systems. If bit TCSEIRNC in this byte is on, there is no communication between System A and System B. This is because of one of the following:

   - System B has not started interregion services
   - System A is out of service with respect to System B
   - System B is out of service with respect to System A

   If bit TCSEIRNC is off, a session should exist.
3. Inspect the primary and secondary session(s) between the systems. The first primary session is pointed to by field TCSEVC1 in the TCTSE, and the first secondary session is pointed to by TCSEVC2. If System A initiated the session, look at secondary sessions, otherwise look at primary sessions.

   Each session is defined by a TCTTE. The field TCTESCCB in TCTTE is zero if the session is not connected to the other system, otherwise it contains the address of the subsystem connection control block (SCCB) that the interregion SVC routine uses to represent that end of the connection.

   Assuming the session is connected, TCTTECA is zero if no task is using the session. Otherwise, TCTTECA points to the TCA of the task that uses the session.

   The protocol for interregion data transfer is similar to that for VTAM SNA data flow control. Field TCTEIRF1 contains information on the state of the session, field TCTESBRS gives the bracket status, field TCTESRHI is the inbound request header, and field TCTESRHO is the outbound request header.

   The field TCTENIBA points to the TCTTE extension for the NIB descriptor. Within this TCTTE extension, TCTEPSQ contains the primary name, and TCTESSQ contains the secondary name. Thus a session in System A can be related to a session in System B.

© Copyright IBM Corp. 1979, 1999

Part 3. Using traces and dumps in problem determination

Part 3 contains:

Chapter 14. Using traces in problem determination 199
Normal CICS tracing 207
CICS exception tracing 220
CICS XRF tracing 221
XRF trace entry types 222
Program check and abend tracing 224
VTAM buffer tracing 224
Using FEPI trace 224

Chapter 15. Using dumps in problem determination 225
Controlling dump action 225
Analyzing dumps 236
Locating the last command or statement 241
Locating program data 242
Storage freeze 252
Using FEPI dump 252

Chapter 16. The global trap/trace exit 253
Establishing the exit 253
Information passed to the exit 254
Actions the exit can take 254
Program check handling 255
Coding the exit 255

© Copyright IBM Corp. 1979, 1999

Chapter 14. Using traces in problem determination

Several types of traces are available to record different aspects of CICS activities. The types of tracing that can be used for CICS systems are:

- "Normal" CICS tracing, is performed by the trace domain at predetermined trace points in CICS code during the regular flow of control, and "user" tracing from applications. You get this when you turn on CICS internal tracing or auxiliary tracing. You can control this type of tracing to suit your needs, except that, when an exception condition is detected by CICS, it always makes an exception trace entry. You cannot turn exception tracing off.
- Tracing from the CICS exit programming interface (XPI), using the TRACE_PUT XPI call from an exit program. You can control this within the exit program, or by enabling and disabling exits.
- CICS XRF tracing, which records CICS XRF-related activities. This is always running if you are operating in a CICS XRF environment.
- Program check and abend tracing, which is used by CICS to record pertinent information when a program check or abend occurs. This is controlled by CICS code.
- VTAM buffer tracing. This is a part of VTAM, but it can be used to record the flow of data between logical units in the CICS environment. You can control this type of tracing to meet your needs.

This chapter starts with a discussion of formatting and interpreting trace entries. This is followed by the characteristics of the various types of tracing in the sections:

- "Normal CICS tracing" on page 207
- "CICS exception tracing" on page 220
- "CICS XRF tracing" on page 221
- "Program check and abend tracing" on page 224
- "VTAM buffer tracing" on page 224

Formatting and interpreting trace entries

Before you can look at the trace entries that have been sent to the various trace destinations, you need to format the traced data. The way you do this varies depending on the destination. For more details of trace utility programs, see the *CICS Operations and Utilities Guide*.

You can specify **abbreviated** or **extended** trace formatting, to give you either a "one line per entry" trace table, or a "many lines per entry" trace table. The structures of the two types of trace entry are described in later sections.

Internal trace can be formatted in one of two ways:

- From a CICS system dump, using the CICS print dump exit, DFHPD410
- From a transaction dump, using the CICS dump utility program, DFHDU410

Auxiliary trace can be formatted using the CICS trace utility program, DFHTU410. You can control the formatting, and you can select trace entries on the basis of task, terminal, transaction, time frame, trace point ID (single or range), dispatcher task reference, and task-owning domain. This complements the usefulness of auxiliary trace for capturing large amounts of trace data.

© Copyright IBM Corp. 1979, 1999

Most of the time, the abbreviated trace table is the most useful form of trace formatting, as you can quickly scan many trace entries to locate areas of interest. However, you might occasionally want to look at the extended format trace entries, to understand fully the information given in the corresponding abbreviated entries, and to be aware of the additional data supplied with many extended trace entries.

Note: Trace entries can only be formatted selectively by transaction or terminal if the “transaction attach” entry (point ID AP F004, KC level-1) for the transaction is included in the trace data set.

Interpreting extended-format CICS system trace entries

CICS system trace entries made to the internal trace table and to auxiliary trace data sets can be formatted to give the same type of information.

There are two slightly different extended trace entry formats. One (“old-style”) resembles the format used in earlier releases of CICS, and gives FIELD A and FIELD B values. The other (“new-style”) uses a completely new format, described below.

Both types of formatted trace entries always show:

- The **trace point ID**. This is an identifier that indicates where the trace point is in CICS code. In the case of application (AP) domain, the request type field included in the entry is also needed to uniquely identify the trace point. For all other domains, each trace point has a unique trace point ID.

  Its format is always a two-character domain index, showing which domain the trace point is in, then a space, then a four-digit (two-byte) hexadecimal number identifying the trace point within the domain.

  The following are examples of trace point IDs:

  ```
  AP 00EE         trace point X'00EE' in Application Domain
  DS 0005         trace point X'0005' in Dispatcher Domain
  TI 0101         trace point X'0101' in Timer Domain
  ```

- An **interpretation string**, showing:
  - The **module** where the trace point is located
  - The **function** being performed
  - Any **parameters** passed on a call, and any response from a called routine.
- A **standard information string**, showing:
  - The **task number**, which is used to identify a task uniquely for as long as it is in the system. It provides a simple way of locating trace entries associated with specific tasks, as follows:
    - A five-digit decimal number shows that this is a trace entry for a task with a TCA, the value being taken from field TCAKCTTA of the TCA.
    - A three-character non-numeric value in this field shows that the trace entry is for a system task. You could, for example, see “III” (initialization), or “TCP” (terminal control).
    - A two-character domain index in this field shows that the trace entry is for a task without a TCA. The index identifies the domain that attached the task.

  - The **kernel task number** (KE_NUM), which is the number used by the kernel domain to identify the task. The same KE_NUM value for the task is shown in the kernel task summary in the formatted system dump.
  - The **time** when the trace entry was made.
  - The **interval** that elapsed between this and the previous trace entry, in seconds.

  The standard information string gives two other pieces of useful information:

  - The address of the **VSE subtask** (field TCB) that is in use for this task. This field can help you in comparing a CICS trace with the corresponding VSE trace.
  - The **return address** (field RET), passed in Register 14 to a called routine. This field helps by showing what invoked the module that is making this trace entry.

- A number of **data fields** containing information relevant to the function being performed.

  For old-style trace points, these are shown as fixed length (4-byte) FIELD A and FIELD B values in the same line as the interpretation string. Both the hexadecimal data values and any printable EBCDIC characters that they represent are shown.

  For new-style trace points, 1–7 variable-length data fields can be given. They are shown immediately below the standard information line. Any printable EBCDIC characters represented by byte values in the data fields are shown on the right of the dump.

  Some of the data fields in the new trace entries contain material intended for use by IBM support personnel and you cannot interpret them directly. However, there is enough information to resolve user errors and for IBM support personnel to resolve most system errors in the interpretation string for the entry. Trace entries from old-style trace points also include a **request type field** (REQ), which gives the same information as the two-byte request field in the formatted trace entries of CICS/VSE® Version 2.

  Some old-style trace entries also have a RESOURCE field. When provided, it is usually the name of a resource associated with the request being traced. For example, for program control requests, it is the program name.

Figure 27 on page 202 shows a trace entry made from an old-style trace point. Its trace point ID is AP 00E1, corresponding to trace ID X'E1' of CICS/VSE Version 2.

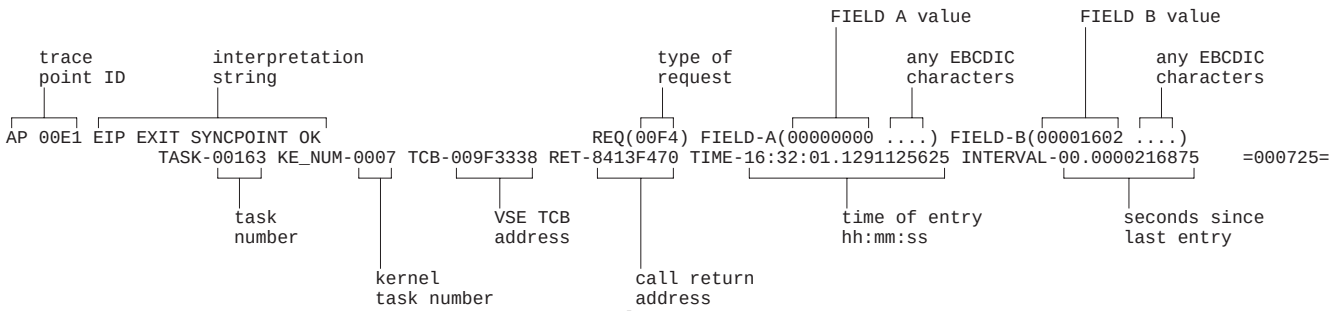


Figure 27. Example of the extended format for an old-style trace entry

You can probably see the resemblance between this type of format and the CICS/VSE Version 2 equivalent it replaces.

The following is an explanation of the old-style trace entry:

- AP 00E1 shows that this trace entry was made from trace point X'00E1' in the application domain.

  Note that although all old-style trace points are in AP domain, not all AP domain trace points are old-style. Some are new trace points, and they have a similar format to that shown in Figure 28 on page 203. In general, old-style trace points have values less than or equal to X'00FF' and new AP-domain trace points have values greater than or equal to X'0200'

- The interpretation string 'EIP EXIT SYNCPOINT OK' gives information about what was going on at the time the trace entry was made.
  - EIP identifies the module where the trace point is located, in this case DFHEIP.
  - EXIT shows that the trace entry was written on completion of processing a request.
  - SYNCPOINT shows the type of function requested.
- REQ(00F4) represents the "request type" of the trace format of CICS/VSE Version 2. In this example, byte 1 bits 0–3 (X'F') show that the trace entry is made on exit from the request.
- FIELD-A and FIELD-B contain the same data as FIELD A and FIELD B in the old-style format.

  FIELD-A bytes 0–3 would contain the secondary response, EIBRESP2. FIELD-B bytes 0–1 would contain the condition number, EIBRESP. In this example, both are zero, indicating that no error response has been returned. FIELD-B bytes 2–3 contain the command code, EIBFN. In this example, this is X'1602', showing that the EXEC CICS command was SYNCPOINT.
- The standard information string shows:
  - The task number for the currently running task is 00163. Any trace entries having the same task number would have been made while this task was running.
  - The kernel task number for the task is 0007. If you were to take a system dump while this task was in the system, you could identify the task in the kernel summary information from this number.

– The time when the trace entry was made was 16:32:01.1291125625.

– The interval that elapsed between this and the preceding trace entry was 00.0000216875 seconds.

Look now at Figure 28, which shows a new-style trace entry.

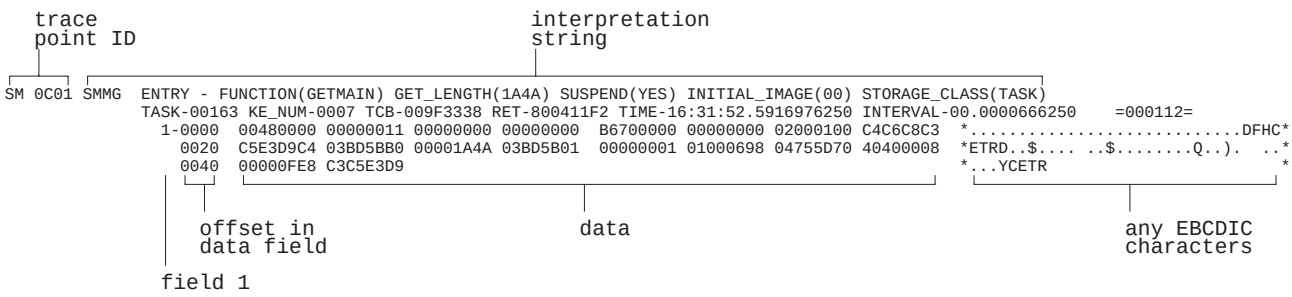


Figure 28. Example of the extended format for a new-style trace entry

The following is an explanation of the new-style trace entry:

- SM 0C01 shows that this trace entry was made from trace point X'0C01' in the storage manager domain.
- The interpretation string provides the following information:
  - SMMG tells you the trace call was made from within module DFHSMMG.
  - ENTRY FUNCTION(GETMAIN) tells you the call was made on entry to the GETMAIN function.
  - GET_LENGTH(1A4A) SUSPEND(YES) INITIAL_IMAGE(00) STORAGE_CLASS(TASK) tells you the parameters associated with the GETMAIN call, as follows:
    - The request is for X'1A4A' bytes of storage.
    - The task is to be suspended if the storage is not immediately available.
    - The storage is to be initialized to X'00'.
    - The storage class is TASK.
- The standard information string gives you this information:
  - The task currently running is task number X'00163'.
  - The kernel task number for the task is 0007.
  - The time when the trace entry was made was 16:31:52.5916976250.
  - The time that elapsed between this and the preceding trace entry was 00.0000666250 seconds.
- The data that is displayed below the standard information was taken from only one data area.

  The *CICS Trace Entries* manual contains details of trace point ID SM 0C01. The data area is the SMMG parameter list.

  Relevant information is formatted from the data area and appears in the trace entry interpretation string.

  Note that information about some data areas is intended for use by IBM support personnel, and this means that details of their format and contents might not be available to you. If you reach a point at which you are certain that

you cannot continue the problem determination process because you do not have access to information about a data area, you need to contact your IBM Support Center. They may not necessarily tell you the format and contents, but they investigate the problem in the usual way, as described in Chapter 17, “IBM program support” on page 259.

Interpreting abbreviated-format CICS system trace entries

Abbreviated-format CICS trace entries contain much of the information present in the corresponding extended-format trace entries, and they are often sufficient for debugging purposes. There is a one-to-one correspondence between the trace entry numbers for the abbreviated and extended trace entries, so you can easily identify the trace entry pairs.

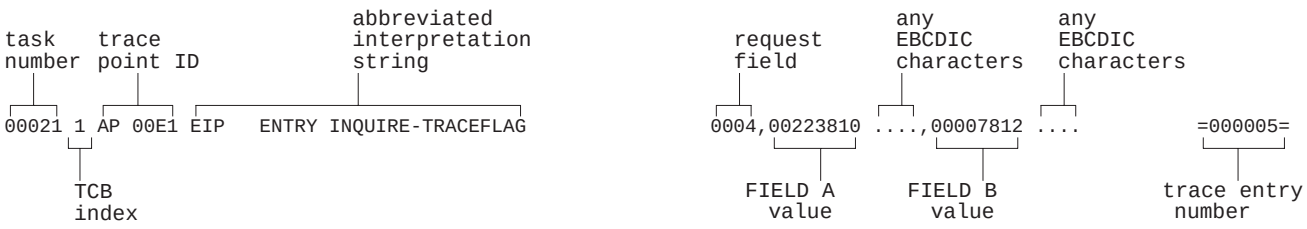
Abbreviated trace entries show a “TCB index” instead of a VSE subtask control block address. The TCB index has a value of 1, 2, or 3, and it indicates when CICS task execution switches from one VSE subtask to another.

The first abbreviated trace entry always has a value of 1 for the TCB index. When execution switches to another subtask, the value is shown as 2. When execution switches again, the value shown is either 1 if the previous subtask is used, or 3 if the remaining subtask is used.

A value of 3 for the TCB index appears only when the FEPI function is used.

There is no absolute correlation between the type of VSE subtask (quasi-reentrant, resource owning, or FEPI) and the value of the TCB index. A value of 1 is always shown for the TCB that happens to be in use when the first trace entry is made.

Figure 29 gives an example of the abbreviated format for an old-style trace entry.



Note: For some trace entries, an 8-character resource field appears to the right of FIELD B.

Figure 29. Example of the abbreviated format for an old-style trace entry

Abbreviated old-style trace entries are easy to interpret, as you can readily identify the REQ, FIELD A and FIELD B fields. Note that some such trace entries also include a RESOURCE field.

For ZCP trace entries, FIELD B (which contains the TCTTE address) is printed twice on each line. This allows both sides of the output to be scanned for the terminal entries on an 80-column screen without having to scroll left and right.

Figure 30 on page 205 gives an example of the abbreviated format for a new-style trace entry.

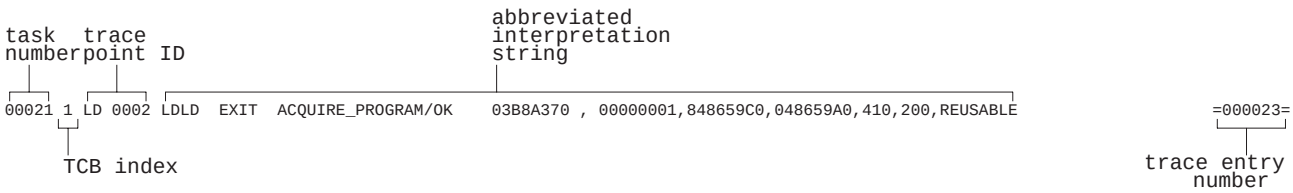


Figure 30. Example of the abbreviated format for a new-style trace entry

Abbreviated-format new-style trace entries are less readily interpreted, because the parameters in the interpretation string are not identified by name. If you are not familiar with the parameters included in the trace entry, you need to look at the corresponding extended-format trace entry to find out what they are. Figure 31 shows the corresponding extended-format trace entry.

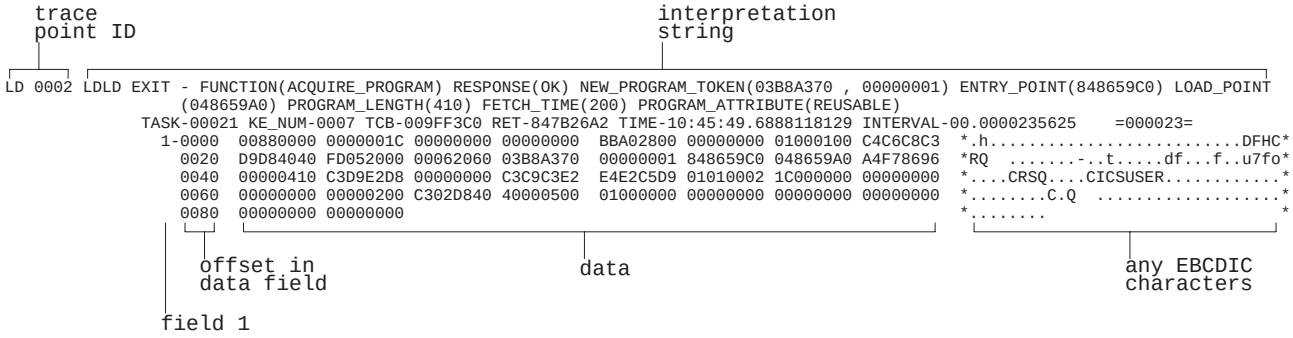


Figure 31. Example of the corresponding extended-format trace entry

LD 0002 shows that this trace entry was made from trace point X'0002' in the loader domain.

The interpretation string provides this information:

- LDLD tells you the trace call was made from within module DFHLDLD.
- EXIT FUNCTION(ACQUIRE_PROGRAM) tells you the call was made on exit from the ACQUIRE_PROGRAM function.

The standard information string gives you this information:

- The task currently running has a task number of 00021.
- The kernel task number for the task is 0007.
- The time when the trace entry was made was 10:45:49.6888118129.
- The time that elapsed between this and the preceding trace entry was 00.0000235625 seconds.

The data displayed below the standard information was taken from only one data area. If you look in the *CICS Trace Entries* manual for details of trace point ID LD 0002, you will see that the data area is the LDLD parameter list.

Interpreting user trace entries

User trace entries have point IDs in the range AP 0000 through AP 00C2, the numeric part of the point ID being specified in the application.

Extended format user trace entries show a user-defined resource field, and a user-supplied data field that can be up to 4000 bytes in length. A typical extended-format entry is shown in Figure 32.

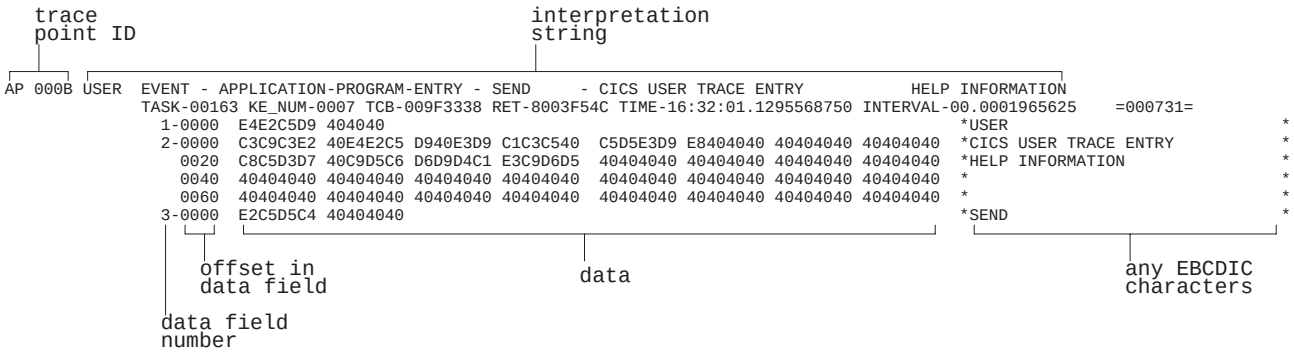


Figure 32. Example of the extended format for a user trace entry

The interpretation string for the entry contains the string "APPLICATION-PROGRAM-ENTRY", to identify this as a user trace entry, and the resource field.

There are three data fields on an extended-format user trace entry:

1. The character string "USER".
2. User data from the area identified in the FROM parameter of the trace command.
3. The resource field value identified in the RESOURCE parameter of the trace command.

The abbreviated-trace entry corresponding to the extended trace entry of Figure 32 is shown in Figure 33.

Figure 33. Example of the abbreviated format for a user trace entry

Abbreviated-format trace entries show the user resource field in the interpretation string. There is also an optional user data field that contains as much user-specified data as can be fitted into the line. If successive user trace entries have the same resource field value, but different data field values, you might need to refer to the corresponding extended trace entries to assess their significance.

Normal CICS tracing

"Normal" CICS tracing is handled by trace domain. It can be used to trace the flow of execution through CICS code, and through your applications as well. For programming information about how to make trace calls from within your own programs, refer to the description of the EXEC CICS ENTER TRACENUM command in the *CICS Application Programming Reference* manual. You can see what functions are being performed, what parameters are being passed, and the values of important data fields at the time trace calls are made. This type of tracing is also used for "first failure data capture", if an exception condition is detected by CICS.

The principal feature of this type of CICS tracing is the control you have over the amount of system tracing that is done. You can select tracing by transaction, by CICS component, and by level of detail.

You can select internal tracing or auxiliary tracing on their own, or combine them so that they are active at the same time. Your choice has no effect on the selectivity with which system tracing is done, but each type of tracing has a set of characteristic properties. These properties are described in "CICS internal trace" on page 216, and "CICS auxiliary trace" on page 217.

Trace points and trace levels

Trace points are included at specific points in CICS code; from these points, trace entries can be written to any currently selected trace destination. All CICS trace points are listed in alphanumeric sequence in the *CICS Trace Entries* manual.

Some trace points are used to make exception traces when exception conditions occur, and some are used to trace the mainline execution of CICS code. Trace points of the latter type each have an associated "level" attribute. The value of this attribute depends on where the trace point is, and the sort of detail it can provide on a trace call.

Trace levels can, in principle, vary in value in the range 1–32, but in practice nearly all mainline trace points have a trace level of 1 or 2.

Level-1 trace points are designed to give you enough diagnostic information to fix "user" errors. The following is a summary of where they are located, and a description of the information they return:

- On entry to, and exit from, every CICS domain. The information includes the domain call parameter list, and data whose address is contained in the parameter list if it is necessary for a high-level understanding of the function to be performed.
- On entry to, and exit from, major internal domain functions. The information includes parameters passed on the call, and any output from the function.
- Before and after calls to other programs, for example, VTAM. The information includes what request is to be made, the input parameters on the request, and the result of the call.
- At many of the points where trace calls were made in CICS/VSE Version 2. The type of information is the same as for that release.

Level-2 trace points are situated between the level-1 trace points, and they provide information that is likely to be more useful for fixing errors within CICS code. You probably will not want to use level-2 trace points yourself, unless you are requested to do so by IBM support staff after you have referred a problem to them.

Level-3 trace points and above are reserved for special cases. Very few components have trace points higher than 2, and they are only likely to be of use to IBM support staff.

You can select how much CICS system tracing is to be done on the basis of the trace level attributes of trace points. You can make your selection independently for each CICS component, and you can also vary the amount of tracing to be done for each task. This gives you control over what system tracing is done.

Note: In the storage manager component (SM), two levels of tracing, level 3 and level 4, are intended for IBM field engineering staff. These trace levels take effect only if specified in system initialization parameters and modify the internal SM operation for CICS subpools as follows:

SM level 3 trace The quickcell mechanism is deactivated. Every CICS subpool, regardless of quickcelling requirements, will issue domain calls for GETMAIN and FREEMAIN services, and these calls will be traced.

SM level 4 trace Subpool element chaining on every CICS subpool is forced. Every CICS subpool, regardless of element chaining requirements, will use element chaining.

A significant performance overhead is introduced into your CICS system if these storage manager trace levels are selected. Note that specifying SM=ALL activates SM trace levels 1, 2, 3, and 4.

Specifying the system tracing you want

For each transaction, you can specify whether *standard* tracing or *special* tracing is to be done, or whether tracing is to be suppressed for that transaction altogether.

For each component, you can specify two sets of trace level attributes. The trace level attributes define the trace point IDs to be traced for that component when standard task tracing is being done and when special task tracing is being done, respectively.

If you are running a test region, you probably have background tracing most of the time. In this case, the default tracing options (standard tracing for all transactions, and level-1 trace points only in the standard set for all components) probably suffice. All you need do is to enable the required trace destinations and set up any related tracing options. Details are given in “Selecting trace destinations and related options” on page 216.

For a production system, background tracing might incur an unacceptable processing overhead. If you find this to be so, you are recommended to set up tracing so that exception traces *only* are recorded on an auxiliary trace data set. There need be no other tracing overhead, and you can be sure that the exception trace will be preserved even when the event invoking the trace does not cause a system dump to be taken. For details, see “CICS exception tracing” on page 220.

When specific problems arise, you can set up special tracing to help you focus on only the relevant tasks and components. A scheme for specifying the tracing you need is outlined in Figure 34 on page 209.

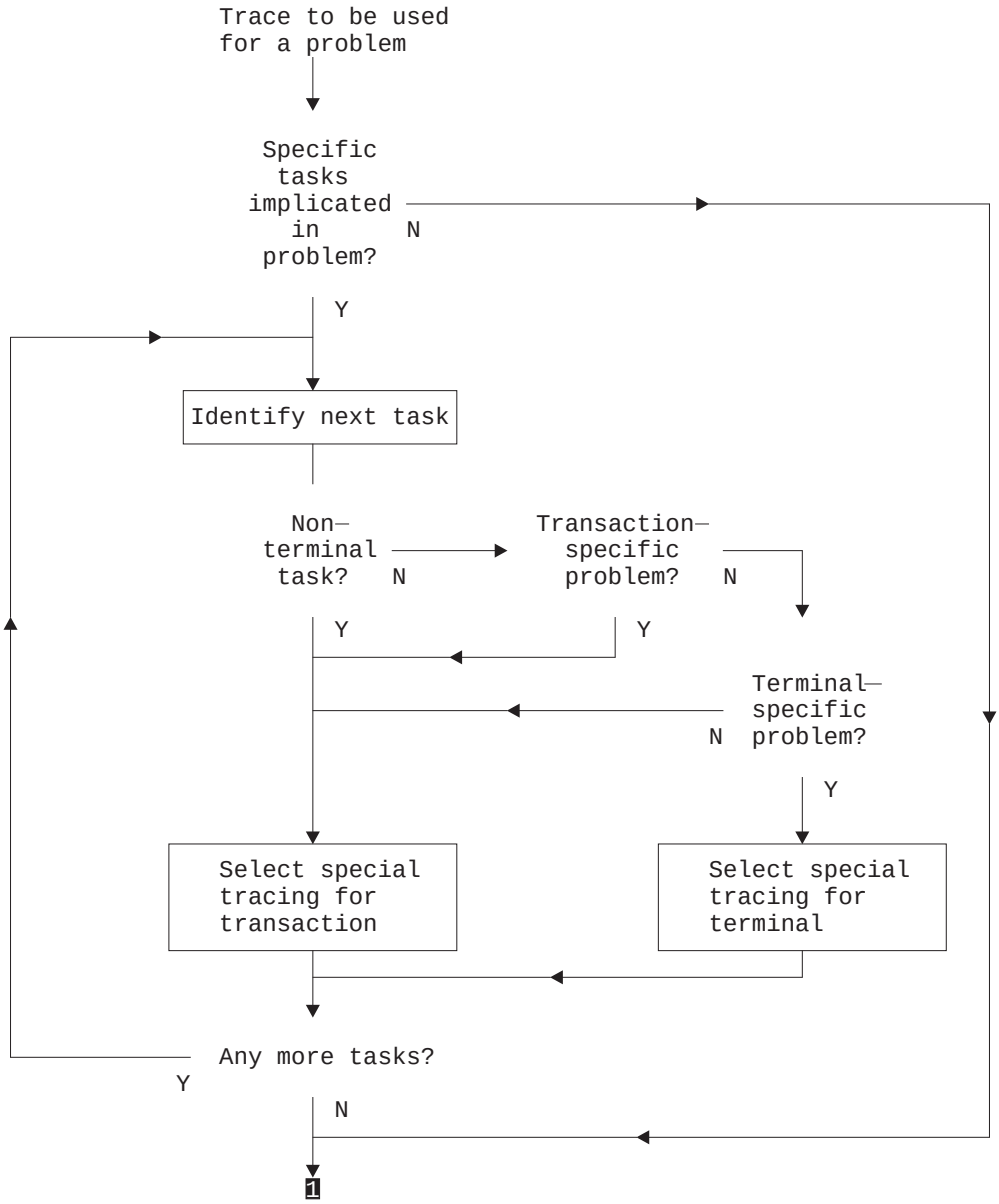


Figure 34 (Part 1 of 3). Outline scheme for setting up special tracing for problems

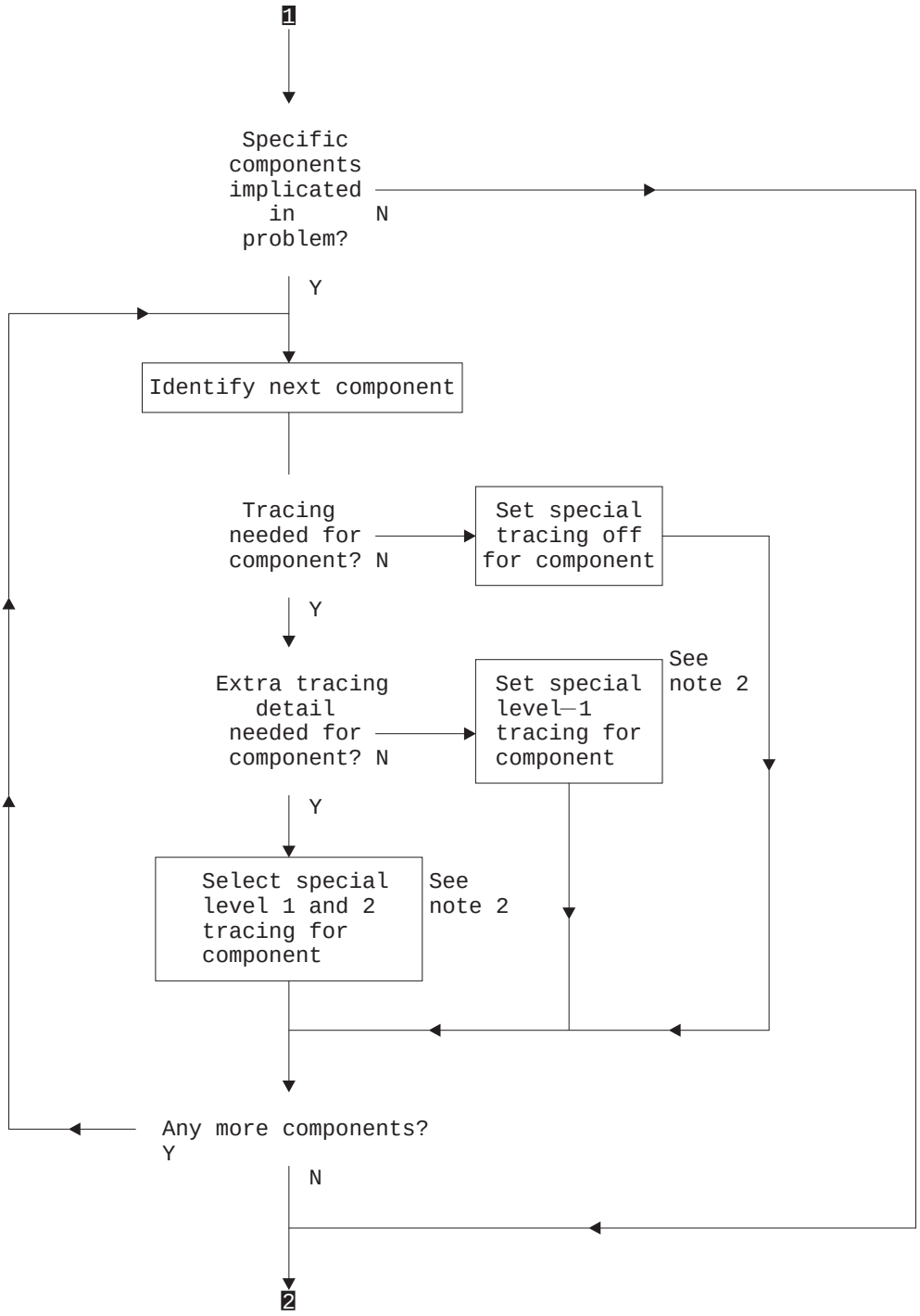


Figure 34 (Part 2 of 3). Outline scheme for setting up special tracing for problems

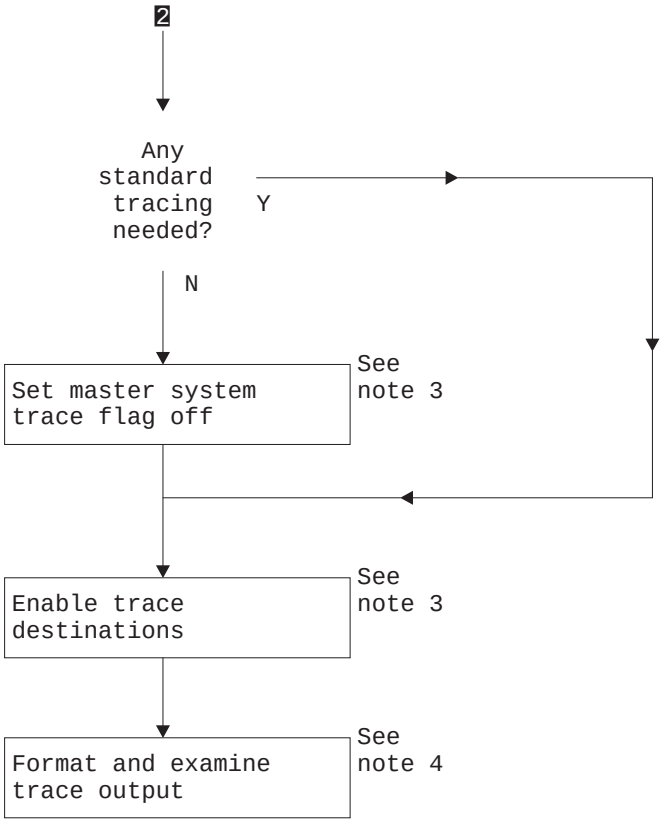


Figure 34 (Part 3 of 3). Outline scheme for setting up special tracing for problems

Notes:

1. See “Tracing for selected tasks”
2. See “Selecting component tracing” on page 214
3. See “Setting the tracing status” on page 218
4. See “Formatting and interpreting trace entries” on page 199; “Interpreting extended-format CICS system trace entries” on page 200; and “Interpreting abbreviated-format CICS system trace entries” on page 204.

Tracing for selected tasks

You can select which tasks are to have standard tracing, which are to have special tracing, and which are to have tracing suppressed. If you specify standard tracing for a task, trace entries are made at all the trace points in the standard set. If you specify special task tracing, you get trace entries at all the trace points in the special set. If you suppress tracing for a task, you do not get any tracing done (except exception tracing) when that task is running.

For transactions that run at terminals, a task is considered to be an instance of a transaction run at a specific terminal. By defining the type of tracing you want by transaction and terminal, you automatically define what task tracing is to be done.

For non-terminal transactions, a task is just an instance of the transaction. The type of tracing you define for the transaction alone defines the type of task tracing that is to be done.

The type of task tracing you get for the various combinations of transaction tracing and terminal tracing is summarized in the truth table shown in Figure 35 on page 212.

OPTION on TRANSACTION	OPTION on TERMINAL	
	standard tracing	special tracing
tracing suppressed	SUPPRESSED	SUPPRESSED
standard tracing	STANDARD	SPECIAL
special tracing	SPECIAL	SPECIAL

Figure 35. The combination of task trace options

You can set up the task tracing you want using the CETR transaction, with the screen shown in Figure 36. Type in the transaction ID or the terminal ID or the netname for the terminal, together with the appropriate tracing.

The status can be any one of STANDARD, SPECIAL, or SUPPRESSED for the transaction, and either STANDARD or SPECIAL for the terminal.

This screen can also be used to set up certain other terminal tracing options. You can select ZCP tracing for a named terminal (trace point ID AP 00E6).

```
CETR                  Transaction and Terminal Trace

Type in your choices.
Item                             Choice    Possible choices
Transaction ID             ===>            Any valid 4 character ID
Transaction Status         ===>            STandard, SPecial, SUppressed

Terminal ID                ===>            Any valid Terminal ID
Netname                    ===>            Any valid Netname
Terminal Status            ===>            STandard, SPecial

Terminal ZCP Trace         ===>            ON, OFf

When finished, press ENTER.

PF1=Help      3=Quit     9=Error List
```

Figure 36. CETR screen for specifying standard and special task tracing

The CETR transaction can, for example, help you to get standard tracing for a transaction when it is run at one terminal, special tracing when it is run at a second terminal, and no tracing at all when it is run at any other terminal.

Notes:

1. You can turn standard tracing off for *all* tasks by setting the master system trace flag off. You can do this with the CETR transaction, using the screen shown in Figure 39 on page 218, or you can code the SYSTR=OFF system initialization parameter. However, any special task tracing will continue—it is not affected by the setting of the system master trace flag.
2. If you run with standard tracing turned off and you specify levels of tracing for the required components under the "Special" heading in the “Components Trace Options” screen shown in Figure 38 on page 215, you can use CETR to trace a single transaction. To do this, specify the transaction ID and a transaction status of SPECIAL, on the screen shown in Figure 36 on page 212.

The tracing logic used by CICS

The logic used by CICS to decide whether a trace call is to be made from a trace point is shown in Figure 37. It is assumed that at least one trace destination is STARTED.

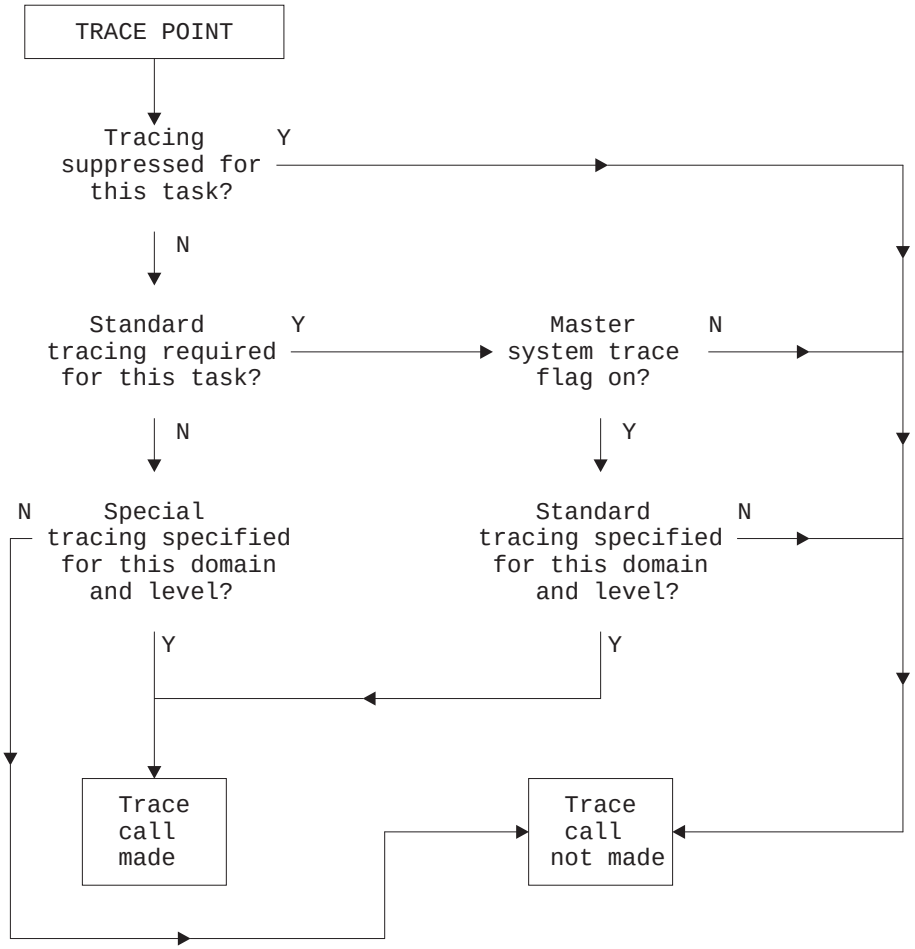


Figure 37. Logic used to determine if a trace call is to be made from a trace point

Selecting component tracing

"Component names and abbreviations" lists the components for which you can select trace levels for standard and special tracing. You can reference this list online through CETR, by pressing PF1 on the component screen (see Figure 38 on page 215).

Decide for each component the trace levels to be used for both standard and special tracing. You can define this either during system initialization, or online using the CETR transaction.

Note: The component codes BF, BM, CP, DC, DI, EI, FC, IC, IS, JC, KC, PC, RC, SC, SP, SZ, TC, TD, TS, and UE are subcomponents of the AP domain. The corresponding trace entries are produced with a point ID of AP nnnn.

For example, trace point AP 0471 is a file control level-1 trace point and AP 0472 is a file control level-2 trace point. These trace points are produced only if the trace setting for the FC component is "(1,2)" or "ALL". The component code AP is used for trace points from the AP domain that do not fall into any of the subcomponent areas listed above.

Component names and abbreviations

Code	Component name'
AP	Application domain'
BF	Built-in function'
BM	Basic mapping support'
CP	Common programming interface'
DC	Dump compatibility layer'
DD	Directory manager domain'
DI	Batch data interchange'
DM	Domain manager domain'
DS	Dispatcher domain'
DU	Dump domain'
EI	Exec interface'
EX	External CICS interface'
FC	File control'
GC	Global catalog domain'
IC	Interval control'
IS	ISC or IRC'
JC	Journal control'
KC	Task control'
KE	Kernel'
LC	Local catalog domain'
LD	Loader domain'
LM	Lock domain'
ME	Message domain'
MN	Monitoring domain'
PA	Parameter domain'
PC	Program control'
PG	Program manager'
RC	Report Controller'
SC	Storage control'
SM	Storage manager domain'
SP	Syncpoint'
ST	Statistics domain'

SZ Front End Programming Interface'
TC Terminal control'
TD Transient data'
TI Timer domain'
TR Trace domain'
TS Temporary storage'
UE User exit interface'
US User domain'
XM Transaction manager'
XS Security manager'

Defining component tracing at system initialization: You can code any of the following CICS system initialization parameters to define component tracing:

- SPCTR, to indicate the level of special tracing required for CICS as a whole.
- SPCTRxx, where xx is one of the two-character component identifiers that specify the level of special tracing you require for a particular CICS component (see “Component names and abbreviations” on page 214).
- STNTR, to indicate the level of standard tracing required for CICS as a whole.
- STNTRxx, where xx is one of the two-character component identifiers that specify the level of standard tracing you require for a particular CICS component (see “Component names and abbreviations” on page 214).

For more information about system initialization parameters, see the *CICS System Definition Guide*.

Defining component tracing when the CICS system is running: You can use the CETR transaction to define component tracing dynamically on the running CICS system.

Figure 38 shows you an example Component Trace Options screen. To make changes, overtype the settings shown on the screen, and then press ENTER.

```
CETR                    Component Trace Options

Overtype where required and press ENTER.                      PAGE 1 OF 2
Component Standard                          Special
--------  --------------------------------  ----------------------------
  AP      1                                 1-2
  BF      1                                 OFF
  BM      1                                 OFF
  CP      1                                 1-2
  DC      1                                 OFF
  DD      1                                 1-2
  DI      1                                 1
  DM      1                                 1-2
  DS      1                                 1-2
  DU      1                                 1-2
  EI      1                                 1
  FC      1                                 1-2
  GC      1                                 1-2
  IC      1                                 1
  IS      1                                 OFF
  JC      1                                 1
  KC      1                                 1

PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Change
```

Figure 38. CETR screen for specifying component trace options

With the settings shown, trace entries are made as follows:

- With standard task tracing in effect, from level-1 trace points of all the components listed.
- With special task tracing in effect:
  - From level-1 trace points only for components DI, EI, IC, JC and KC
  - From both level-1 and level-2 trace points for components AP, CP, DD, DM, DS, DU, FC, and GC

  No special task tracing is done for components BF, BM, DC, and IS.

Selecting trace destinations and related options

The trace destinations you select and the options you define for the destinations depend on your specific problem determination requirements. You can opt to run CICS internal tracing or CICS auxiliary tracing, or you can elect to have both active at the same time. See in “CICS internal trace” and “CICS auxiliary trace” on page 217 for more information. Decide how much trace data you need to capture, and whether you want to integrate CICS tracing with tracing done by other programs.

You can control the status and certain other attributes of the various types of CICS tracing either dynamically, using the CETR transaction, or during system initialization, by coding the appropriate system initialization parameters.

CICS internal trace

You can select a status of STARTED or STOPPED for internal trace. If you select STARTED, any trace calls that are made cause trace entries to be directed to the internal trace table. This occupies a sequence of pages in virtual storage above the 16MB line, in VSE GETVIS storage.

The internal trace table has a minimum size of 16KB, and a maximum size of 1048576KB. The table is extendable from 16KB in 4KB increments. You can change the size of the table dynamically, while CICS is running, but if you do so you lose all of the trace data that was present in the table at the time of the change. If you want to keep the data *and* change the size of the table, take a system dump before you make the change.

The internal trace table wraps when it is full. When the end of the table is reached, the next entry to be directed to the internal trace entry goes to the start, and overlays the trace entry that was formerly there. In practice, the internal trace table cannot be very big, so it is most useful for background tracing or when you do not need to capture an extensive set of trace entries. If you need to trace CICS system activity over a long period, or if you need many entries over a short period, one of the other trace destinations is likely to be more appropriate.

Note that the internal trace table is always present in virtual storage, whether you have turned internal tracing on or not. The reason is that the internal trace table is used as a destination for trace entries when CICS detects an exception condition. Other trace destinations that are currently selected get the exception trace entry as well, but the entry always goes to the internal trace table even if you have turned tracing off completely. This is so that you get “first failure data capture”.

CICS auxiliary trace

CICS auxiliary trace entries are directed to one of two auxiliary trace data sets, DFHAUXT and DFHBUXT. These are CICS-owned SAM data sets, and they must be created before CICS is started. They cannot be redefined dynamically.

You can use the AUXTR system initialization parameter to turn CICS auxiliary trace on or off during CICS initialization.

You can select a status of STARTED, STOPPED, or PAUSED for CICS auxiliary trace dynamically using the CEMT SET AUXTRACE command or the CETR transaction. These values reflect both the status of the auxiliary trace flag, and the status of the current auxiliary trace data set, in the way shown in Table 30.

Table 30. The meanings of auxiliary trace status values

Auxiliary tracing status	Auxiliary trace flag	Auxiliary trace data set
STARTED	ON	OPEN
STOPPED	OFF	CLOSED
PAUSED	OFF	OPEN

When you first select STARTED for CICS auxiliary trace, any trace entries are directed to the initial auxiliary trace data set. If CICS terminated normally when auxiliary trace was last active, this is the auxiliary trace data set that was not being used at the time. Otherwise, it is the DFHAUXT data set. If you initialize CICS with auxiliary trace STARTED, DFHAUXT is used as the initial auxiliary trace data set.

What happens when the initial data set is full depends on the status of the auxiliary switch (AUXTRSW). This can have a value of NO, NEXT, or ALL, and you can define it either using the AUXTRSW system initialization parameter, or dynamically using the CEMT SET AUXTRACE command or the CETR transaction.

AUXTRSW value — **Meaning**

NO When the initial data set is full, no more auxiliary tracing is done.

NEXT When the initial data set is full, the other data set then receives the next trace entries. However, when that one is full, no more trace data is written to auxiliary trace.

ALL Auxiliary trace data is written alternately to each data set, a switch being made from one to the other every time the current one becomes full. This means that trace entries already present in the trace data sets start getting overwritten when both data sets become full for the first time.

The advantage of using auxiliary trace is that you can collect large amounts of trace data, if you initially define large enough trace data sets. For example, you might want to do this to trace system activity over a long period of time, perhaps to solve an unpredictable storage violation problem.

A time stamp is included in the header line of every page of abbreviated auxiliary trace output to help match external events with a particular area of the trace, and thus help you to find the trace entries that are of interest.

Setting the tracing status

You can set the system tracing status by coding the appropriate system initialization parameters, and you can also set it dynamically, using the CETR transaction. The CETR transaction enables you to make changes in response to contingencies, as they arise.

Setting the tracing status at system initialization

These are the parameters that you can use to set up tracing status at system initialization:

AUXTR
To specify whether auxiliary trace is to be activated at system initialization.

AUXTRSW
To define the auxiliary switch status.

INTTR
To specify whether internal tracing is to be activated at system initialization.

SYSTR
To set the master system trace flag on or off.

TRTABSZ
To specify the size of the internal trace table.

USERTR
To set the master user trace flag on or off. It must be on if user trace calls are to be made from your applications.

For more details of these system initialization parameters, see the *CICS System Definition Guide*.

Setting the tracing status using CETR

You can use the CICS trace control transaction (CETR) to set the tracing status. Figure 39 shows you the CETR screen you can use to set the tracing status dynamically.

```
CETR                   CICS Trace Control Facility
Type in your choices.
Item                              Choice       Possible choices
Internal Trace Status       ===>  STOPPED      STArted, STOpped
Internal Trace Table Size   ===>  0016 K       16K - 1048576K

Auxiliary Trace Status      ===>  STARTED      STArted, STOpped, Paused
Auxiliary Trace Dataset     ===>  B            A, B
Auxiliary Switch Status     ===>  ALL          NO, NExt, All

Master System Trace Flag    ===>  OFF          ON, OFf
Master User Trace Flag      ===>  OFF          ON, OFf

When finished, press ENTER.

PF1=Help      3=Quit      4=Components      5=Ter/Trn     9=Error List
```

Figure 39. Tracing options shown on a CETR screen

In this example, internal tracing status is STOPPED, and so regular tracing is not directed explicitly to the internal trace table. However, note that the internal trace table is used as a buffer for the other trace destinations, so it always contains the most recent trace entry, if at least one trace destination is STARTED. The internal trace table is also used as a destination for exception trace entries, which are made whenever CICS detects an exception condition. If such a condition were detected when the options shown in this example were in effect, you would be able to find the exception trace entry in the internal trace table.

The internal trace table size is 16KB, which is the minimum size it can be. If internal trace were STARTED, the trace table would wrap when it became full.

The current auxiliary trace data set is B, meaning that trace entries are being written to DFHBUXT. The auxiliary switch status is ALL, so a switch would be made to the other auxiliary trace data set whenever one became full.

The master system trace flag is OFF. This means that no standard tracing is done at all, even though standard tracing might be specified for some tasks. However, special task tracing is not affected—the master system trace flag only determines whether standard task tracing is to be done.

You can see the role of the master system trace flag in Figure 37 on page 213.

The master user trace flag is OFF, so no user trace entries can be made from applications. You must set the master user trace flag on before any user trace requests in your programs can be serviced. If it were off, any trace call requests in your programs would be ignored.

The logic used to ensure that trace entries are written to the required destinations is shown in Figure 40.

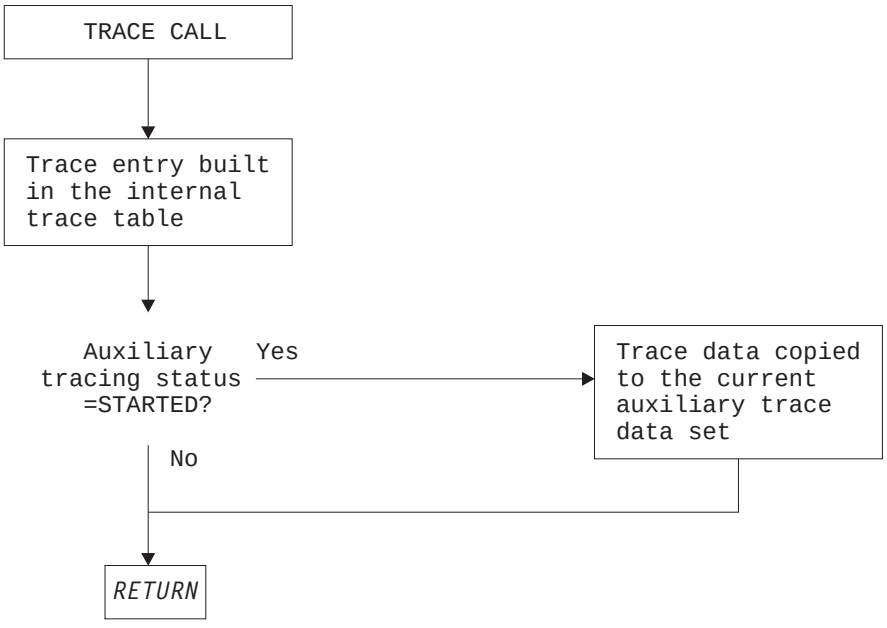


Figure 40. How trace entries are directed to the required destinations

CICS exception tracing

CICS always performs exception tracing when it detects an exception condition (for example, bad parameters on a domain call, or an abnormal response from a called routine). The aim is “first failure data capture”, to record data that might be relevant to the exception as soon as possible after it has been detected.

CICS uses a similar mechanism for both exception tracing and “normal” tracing. Exception trace entries are made from specific points in CICS code, and data is taken from areas that might provide information about the cause of the exception. The first data field in the trace entry is usually the parameter list from the last domain call, because this can indicate the reason for the exception.

The exception trace points do not have an associated “level” attribute, and trace calls are only ever made from them when exception conditions occur.

Exception trace entries are always written to the internal trace table, even if no trace destinations are currently STARTED. That is why there is always an internal trace table in every CICS partition, to make sure there is always somewhere to write exception trace entries. If the other trace destinations are STARTED, the exception trace entries are written there, as well.

You can select tracing options so that exception traces *only* are made to an auxiliary trace data set. This is likely to be useful for production regions, because it enables you to preserve exception traces in auxiliary storage without incurring any general tracing overhead. You need to disable all standard and special task tracing, and enable auxiliary trace:

1. Ensure that special tracing has not been specified for any task.
2. Set the master system trace flag off.
3. Set the auxiliary trace status to STARTED, and the auxiliary trace data set and the auxiliary switch status to whatever values you want.

Exception traces are now made to an auxiliary trace data set, but there is no other tracing overhead.

The format of an exception trace entry is almost identical to that of a normal trace entry. However, you can identify it by the eye-catcher ***EXC*** in the header.

Note: Exception conditions that are detected by VSE—for example, operation exception, protection exception, or data exception—do not cause a CICS exception trace entry to be made directly. However, they do cause a CICS recovery routine to be invoked, and that, in turn, causes a “recovery” exception trace entry to be made.

User exception trace entries

The EXCEPTION option on the EXEC CICS ENTER TRACENUM command enables user programs to write a trace entry to the trace destinations, even when the master user trace flag is off. User exception trace entries are always written to the internal trace table (even if internal tracing is set off), but are written to other destinations only if they are active.

The user exception trace entries that CICS writes are identified by the character string *EXCU in any formatted trace output produced by CICS utility programs. For example, an application program exception trace entry generated by an EXEC CICS ENTER TRACENUM() EXCEPTION command appears in formatted trace output as:

```
USER *EXCU - APPLICATION-PROGRAM-EXCEPTION
```

If you use the exit programming interface (XPI) trace control function to write user trace entries, you can use the DATA1 block descriptor to indicate whether the entry is an exception trace entry. Enter the literal 'USEREXC' in the DATA1 field on the DFHTRPTX TRACE_PUT call to identify an exception trace entry. This is interpreted by the trace formatting utility program as follows:

```
USER *EXCU - USER-EXIT-PROGRAM-EXCEPTION
```

See the *CICS Customization Guide* for programming information about XPI trace control function.

CICS XRF tracing

CICS XRF tracing is always active when you are running with XRF. It is used by the CICS availability manager (CAVM), and you cannot turn it off. However, it only makes about 12 entries every 2 seconds, so the overhead is not great. Note that CICS XRF tracing is quite distinct from the "normal" CICS tracing that can originate from the CAVM, which is identified by trace point IDs AP 00C4 through AP 00C7.

The XRF trace entries are 32 bytes long and are written to a trace table in main storage. The table has a fixed size of 64KB, and it wraps around when it is full.

The table starts with 28 bytes of control information, in the format shown in Table 31.

Table 31. Control information at the start of the XRF trace table

Bytes	Contents
0–15	'*** XRF TRACE **'
16–19	Address of start of trace entries
20–23	Address of end of trace entries
24–27	Address of end of most recent entry

Trace entries are 32 bytes long, and have the format shown in Table 32.

Table 32. Format of an XRF trace entry

Bytes	Contents
0	Type code
1	Subtype
2–3	Process ID of XRF process that made the entry
4–27	Trace data—the format depends on the type or the subtype
28–31	Clock value when entry was made, same format as "normal" CICS trace entries

Process IDs are assigned in order of process ATTACH starting from 1. Some special values are used for processes which are not known to the dispatcher, but which cause trace entries to be made. These are:

Process ID	Function
X'0000'	Initial attach
X'FFFE'	STXIT AB error handling
X'FFFF'	Dispatcher activities.

XRF trace entry types

The entries are as follows:

Table 33 (Page 1 of 3). XRF trace entry types

Module	Type	Subtype	Description
DFHWLGET	1	1	Module entry Bytes: 4-11 Module name 12-15 LIFO allocation address
DFHWLFRE	1	2	Module return Bytes: 4-11 Module name 12-15 LIFO allocation address 16-27 0
DFHWDATT	2	1	XRF process attach Bytes: 4- 7 Process entry point 8-11 Initial data parameter 12-15 Address of STXITPC routine 16-19 Address of STXITAB routine 20-23 Address of attached process XPB 24-27 Process ID attached process XPB
DFHWDISP	2	2	XRF process detach Bytes: 4-27 0
DFHWDISP	2	3	XRF process dispatch Bytes: 4- 7 Address of external ECB waited for (if any) 8-11 Address of internal ECB waited for (if any) 12-15 Awaited broadcast events that were posted 16-19 Broadcast events still posted for this process 20-23 Address of process XPB 24-27 Locks held by this process

Table 33 (Page 2 of 3). XRF trace entry types

Module	Type	Subtype	Description
DFHWDWAT	2	4	XRF process wait (event data) Bytes: 4- 7 Address of external ECB to wait for (if any) 8-11 Address of internal ECB to wait for (if any) 12-15 Broadcast events to wait for (if any) 16-19 Events to be broadcast to all processes 20-23 Events to be reset for this process 24-27 0
DFHWDAT	2	5	XRF process wait (lock data) Bytes: 4- 7 Locks to be freed 8-11 Locks to be acquired 12-19 0 20-23 Locks held by all other processes at time of call 24-27 Locks held by this process at time of call
DFHWDISP	2	6	Dispatcher termination Bytes: 4-27 0
DFHDISP	2	7	Dispatcher issuing operating system WAIT Bytes: 4-19 0 20-23 Address of WAIT list 24-27 Number of ECBs in WAIT list
DFHWDISP	2	8	Dispatcher resume after operating system WAIT Bytes: 4-27 0
DFHWMMT	3	1	Message manager issuing VSAM GET Bytes: 4- 7 RPL address 8-11 RBA of CI to be read 12-27 0
DFHWMMT	3	2	Message manager issuing VSAM PUT Bytes: 4- 7 RPL address 8-11 RBA of CI to be written 12-27 0
DFHWMMT	3	3	Message manager I/O complete Bytes: 4- 7 RPL address 8-11 RBA of CI to be read 12 0 13-15 VSAM feedback information 16-27 0

Table 33 (Page 3 of 3). XRF trace entry types

Module	Type	Subtype	Description
DFHWMQH	4	1	Message manager message received Bytes: 4- 7 0 8-11 Queue name 12-15 Message sequence number 16-19 Address of message block (contains message copy) 20-27 0
DFHWMWR	4	2	Message manager message sent Bytes: 4- 7 0 8-11 Queue name 12-15 Message sequence number 16-19 Message file cycle number 20-23 RBA of this message 24-25 0 26-27 Response to PUTMSG request (WMSRESP)

Program check and abend tracing

Program check and abend tracing is carried out by kernel domain. The kernel records information about program checks and abends, including details of the registers and the PSW at the time of failure.

You cannot format the program check and abend trace information directly, but you can get a summary of its contents in a formatted CICS system dump by specifying dump formatting keyword KE. The information is provided in the form of a storage report for each task that has had a program check or an abend during the current run of CICS.

An example of such a storage report is given in Figure 2 on page 45.

VTAM buffer tracing

VTAM buffer tracing enables you to look at all the data that is passed between logical units on a VTAM communication link. For details of VTAM buffer tracing, see the appropriate manual in the VTAM library.

Using FEPI trace

For information about using trace to solve FEPI problems, see the *CICS Front End Programming Interface User's Guide*.

Chapter 15. Using dumps in problem determination

You have the choice of two different types of CICS dump to help you with problem determination. They are the **transaction dump**, of transaction-related storage areas, and the **CICS system dump**, of the entire CICS partition.

The type of dump to use for problem determination depends on the nature of the problem. In practice, the system dump is often more useful, because it contains more information and you can be reasonably confident that it has captured all the evidence you need to solve your problem.

The CICS system dump is potentially very large, but this need not be a problem. You can leave the data on the system dump data set, or keep a copy of it, and format it selectively as you require.

You can control the dump actions taken by CICS, and also what information the dump output contains.

Controlling dump action

There are two aspects to controlling dump action:

- Setting up the dumping environment, so that appropriate dump action is taken when circumstances arise that cause a dump to be taken.
- Causing a dump to be taken. Both users and CICS can issue requests for dumps to be taken.

Setting up the dumping environment

There are several levels at which the dumping environment can be set up:

- System dumps (apart from CICS kernel domain dumps) can be globally suppressed or enabled at system initialization by using the DUMP system initialization parameter.
- System dumps (apart from CICS kernel domain dumps) can be globally suppressed or enabled dynamically through the CEMT or EXEC CICS SET SYSTEM DUMPING command.
- Transaction dumps can be suppressed or enabled dynamically for individual transactions. This is done by using the EXEC CICS SET TRANSACTION DUMPING system programming command, or by using the DUMP attribute of the RDO TRANSACTION resource definition.
- System dumps (apart from CICS kernel domain dumps) can be suppressed for specific dump codes from a dump domain XDUREQ global user exit program For details of XDUREQ, see the *CICS Customization Guide*
- System dumps (apart from CICS kernel domain dumps) can be suppressed or enabled dynamically for specific system abend codes. You can do this using the CEMT SET SYDUMPCODE command or the EXEC CICS SET SYSDUMPCODE system programming command.
- Transaction dumps can be suppressed or enabled dynamically for specific transaction abend codes. This can be done by using the CEMT SET

© Copyright IBM Corp. 1979, 1999

TRDUMPCODE command or the EXEC CICS SET TRANDUMPCODE system programming command

A dump code defines what action CICS is to take under any of the circumstances in which a dump might be required. Dump codes are kept in one of two **dump tables**, one for "transaction dump codes" and the other for "system dump codes". For details, see "The dump code options you can specify" on page 229.

- Dumps of the builder parameter set at specific stages in the build process of terminal or connection definition can be enabled. This is done using the CSFE ZCQTRACE facility. For details, see "The CSFE ZCQTRACE facility" on page 235.

Events that can cause dumps to be taken

The following events cause dumps to be taken, if the dumping environment allows dumping under the circumstances:

- Explicit requests for dumps from users
- CICS transaction abends
- CICS system abends

On most occasions when dumps are requested, CICS references a dump code that is specified either implicitly or explicitly to determine what action should be taken. Dump codes are held in two dump tables, the transaction dump table and the system dump table.

The ways that you can request dumps

You can issue an explicit request for a dump by using the CEMT transaction, by using an EXEC CICS command, or by using an exit programming interface (XPI) call.

- CEMT PERFORM [DUMP|SNAP] enables you to get a CICS system dump from the master terminal, if system dumping has not been globally suppressed. The system dump code that is needed is supplied by CICS, and it has a specific value of "MT0001".
- You can use EXEC CICS PERFORM DUMP to get a CICS system dump, if system dumping is not globally suppressed. You must specify a system dump code when you use this command and it must be a maximum of 8 characters in length.
- You can use EXEC CICS DUMP TRANSACTION to get a transaction dump. You get a transaction dump even if dumping has been suppressed for the transaction that you identify on the command.

  You must specify a transaction dump code when you use this command and it must be a maximum of 4 characters in length. It could, for example, be TD01.
- You can make a TRANSACTION_DUMP or a SYSTEM_DUMP XPI call from a global user exit (GLUE) program to get a transaction dump or a system dump, respectively. For programming information about these exits, see the *CICS Customization Guide*.

You might use these methods of taking dumps if, for example, you had a task in a wait state, or you suspected that a task was looping. However, these methods are not useful for getting information following a transaction abend or a CICS system

abend. This is because the evidence you need is almost certain to disappear before your request for the dump has been processed.

The occasions when CICS requests a dump

In general, CICS requests a dump when a transaction or CICS system abend occurs. The dumping environment determines whether or not a dump is actually taken. CICS does not take a transaction dump if a HANDLE ABEND is active[4]. The NODUMP option on an EXECS CICS ABEND command, an internal call, or the transaction definition, prevents the taking of a transaction dump.

The following are the occasions when CICS requests a dump:

- CICS requests a transaction dump, and perhaps a system dump, after a transaction abend occurs. There are two cases:
  - You can include the command EXEC CICS ABEND in one of your applications, causing it to abend when some condition occurs. You must specify a four-character transaction abend code on this command, and this is used as the transaction dump code. It could, for example, be 'MYAB'.
  - CICS might cause a transaction to abend, for any of the reasons described in the *VSE/ESA Messages and Codes Volume 3* manual. In this case, the four-character CICS transaction abend code is used as the transaction dump code. It might, for example, be ASRA.
- A CICS system dump could be taken following a CICS system abend. In this situation, the system dump code is often equal to the abend message ID, with the leading letters "DFH" stripped off. In the case of message DFHST0001, for example, the system dump code would be "ST0001". However, in some cases the system dump code cannot be directly related to a CICS message, either because it does not closely resemble the message, or because no message accompanies the event causing the dump to be invoked. For details of these system dump codes, see the *VSE/ESA Messages and Codes Volume 3* manual.
- A CICS system dump can be requested from within the Node Error Program (NEP) when a terminal error is processed for a terminal with no task attached. For information about using NEPs to handle terminal errors, read the entry for message DFHZC3496 in the *VSE/ESA Messages and Codes Volume 3* manual; also read the programming information on NEPs in the *CICS Customization Guide*.
- A CICS system dump can also be requested from the global trap/trace exit. In this case, the system dump code is TR1003.

Enabling system dumps for some CICS messages

There are occasions when you might need diagnostic information for an event that causes a message to be sent, but does not normally cause CICS to take a system dump. You can enable system dumping for some CICS messages in such a situation by adding the dump code (constructed by removing the "DFH" prefix from the message number) to the system dump table, and specifying the SYSDUMP

---

4 To make PL/I on units work, PL/I library routines can issue HANDLE ABEND. This is called an implicit HANDLE ABEND and causes the suppression of transaction dumps.

option. CICS then causes a system dump to be taken when the message is issued.

To determine which messages you can do this for, look in the *VSE/ESA Messages and Codes Volume 3* manual. If the message you are interested in has a 2-character alphabetic component ID after the “DFH” prefix, and it has **either** XMEOUT global user exit parameters or a destination of “Terminal User”, you can use it to construct a system dump code to add to the dump table.

You cannot enable dumping for messages that do not have these characteristics. For example, some messages that are issued early during initialization cannot be used to cause CICS to take a system dump, because the mechanisms that control dumping might not be initialized at that time. Also, you cannot enable dumping for the message domain's own messages (they are prefixed by “DFHME”) where they do not normally cause CICS to take a system dump.

System dump actions with messages DFHAP0001 and DFHSR0001

In the event of a program check or VSE abend in the AP domain or in a user application program, CICS may issue either message DFHAP0001 or DFHSR0001. Message DFHSR0001 is issued by CICS only when storage protection is active; that is, the system initialization parameter STGPROT=YES is specified and the hardware and software environment supports the storage protection facility. CICS determines which of these messages to issue depending on whether or not the program check or VSE abend occurs in code running in user key.

If the code had been running in user key at the time of the program check or VSE abend, CICS issues message DFHSR0001 and takes a system dump with dump code SR0001. Only application programs defined with EXECKEY(USER) run in user key.

If the code had not been running in user key at the time of the program check or VSE abend, CICS issues message DFHAP0001 and takes a system dump with dump code AP0001.

So, if CICS storage protection is active, this mechanism enables you to suppress the system dumps caused by errors in application programs, while still allowing dumps caused by errors in CICS code to be taken. To achieve this, use either a CEMT SET SYDUMPCODE or an EXEC CICS SET SYSDUMPCODE command to suppress system dumps for system dumpcode SR0001:

```
CEMT SET SYDUMPCODE(SR0001)  ADD  NOSYSDUMP
```

If storage protection is not active, the dumps may be suppressed via a suppression of dumpcode AP0001. Note, however, that this suppresses dumps for errors in both application **and** CICS code.

You cannot control AP0001 and SR0001 system dumps by using the DUMP parameter of the RDO TRANSACTION definition. The DUMP parameter of the TRANSACTION definition controls only transaction dumps.

Usually, program checks, or VSE abends caused by an application program, are also followed by an ASRA, ASRB or ASRD transaction abend and a transaction dump. If, in some instances, you want the SDUMP for one of these transaction abends but not the other, specify the one you want by using either a CEMT

TRDUMPCODE or an EXEC CICS TRANDUMPCODE command. For example, specifying:

```
CEMT SET TRDUMPCODE(ASRB)  ADD  SYSDUMP
```

adds an entry to the dump table and ensures that SDUMPs are taken for ASRB abends. However, note that the SDUMP in this instance is taken at a later point than the SDUMP normally taken for system dump code AP0001 or SR0001.

The dump code options you can specify

You can specify what dump action is to be taken by CICS for each individual dump code, either by using a CEMT transaction or by using a system programming command. The options you can specify differ slightly, depending on whether you are defining the action for a transaction dump code or for a system dump code.

- For a **transaction dump code**, you can specify:
  - Whether a transaction dump is to be taken
  - Whether a system dump is to be taken, with or without a transaction dump
  - Whether CICS is to be terminated
  - The maximum number of times the transaction dump code action can be taken during the current run of CICS, or before the count is reset
- For a **system dump code**, you can specify:
  - Whether a system dump is to be taken
  - Whether CICS is to be terminated
  - The maximum number of times the system dump code action can be taken during the current run of CICS, or before the count is reset

Notes:

1. Only a transaction dump code can cause both a transaction dump and a system dump to be taken.
2. If a severe error is detected, the system can terminate CICS even if you specify that CICS is not to be terminated.
3. Values of 0–998 for “maximum times dump code action can be taken” are literal, but a value of 999 (the default) means there is no limit.
4. You cannot suppress CICS kernel domain dumps.

All the options you specify are recorded in the appropriate dump table, and written in the CICS global catalog. Any dump table entries you have changed or created during a CICS run are preserved when CICS is subsequently shut down. They are available during the next run unless CICS is cold started, when they are deleted from the global catalog.

The only circumstances in which dump table additions and changes are lost are:

- When CICS is cold started.
- When the CICS global catalog is redefined, although this is likely to be done only in exceptional circumstances.
- When CICS creates a temporary dump table entry for you, because you have asked for a dump for which there is no dump code in the dump table.

Dump table statistics

CICS maintains the following statistics for each dump table entry:

- The number of times the dump code action has been taken. This count is referred to as the “dumps-taken”. Its value is incremented every time the dump code action is taken, irrespective of what action has been specified in the dump table entry. The current count can be reset to zero by a CEMT transaction, by a system programming command, or by statistical interval expiry.

  If system dumping is globally suppressed:

  – The dumps-taken count for a system dump code is not incremented by dump requests using that dump code, but the dumps-suppressed count is incremented.

  – The dumps-taken count for a transaction dump code specifying a system dump is incremented by dump requests using that dump code. This is so even if the dump code specifies that only a system dump is to be taken.

- For a transaction dump code, the number of transaction dumps that have been taken. The number is incremented every time a transaction dump is taken for this dump code. However, it is not incremented if transaction dumping has been suppressed in this table entry.
- For a transaction dump code, the number of transaction dumps that have been suppressed.
- The number of system dumps that have been taken. The number is incremented every time a system dump is taken for this dump code. It is not incremented if system dumping has been suppressed, either explicitly in the dump table entry or because system dumping has been suppressed globally.
- The number of system dumps that have been suppressed, either explicitly for this dump code or because system dumping has been suppressed globally.

Notes:

1. Dump code statistics are all reset when CICS is shut down—unlike dump code attributes, which are not reset.
2. The following dump code statistics are reset at the end of every statistics collecting interval:
   - Number of transaction dumps taken
   - Number of transaction dumps suppressed
   - Number of system dumps taken
   - Number of system dumps suppressed

The transaction dump table

Table 34 shows some examples of the type of information that might be maintained in the transaction dump table for different transaction dump codes.

Table 34 (Page 1 of 2). Examples of transaction dump table entries

Type of information	Example 1	Example 2	Example 3
Transaction dump code	MYAB	ASRA	AKC3
Take a transaction dump?	YES	NO	NO
Take a system dump?	YES	YES	NO

Table 34 (Page 2 of 2). Examples of transaction dump table entries

Type of information	Example 1	Example 2	Example 3
Shut down CICS?	NO	NO	NO
Maximum times dump code action can be taken	50	30	999
Times dump code action already taken (current count)	0	30	37
Transaction dumps taken	0	0	0
Transaction dumps suppressed	0	30	37
System dumps taken	0	30	0
System dumps suppressed	0	0	37

- **Example 1** shows a transaction dump table entry for transaction dump code MYAB. This is a user-supplied dump code, specified either on an EXEC CICS DUMP TRANSACTION command, or as a transaction abend code on an EXEC CICS ABEND command.

  The table entry shows that when this dump code is invoked, both a transaction dump and a system dump are to be taken, and CICS is not to be terminated. The dump code action can be taken a maximum of 50 times, but the action for this dump code has not been taken since CICS was started or since the current count ("times dump action taken") was reset.

- **Example 2** shows a transaction dump table entry for transaction dump code ASRA. This is a CICS transaction abend code, and this dump table entry is referred to every time a transaction abends ASRA. The entry shows that a system dump only is to be taken for an ASRA abend, and that CICS is not to be terminated. It also shows that the action for this abend code has already been taken the maximum number of times, so no action is taken when another ASRA abend occurs. However, the current count could be reset to 0 dynamically using either a CEMT transaction or a system programming command (SET TRANDUMPCODE or SET SYSDUMPCODE respectively). More system dumps would then be taken for subsequent ASRA abends.

- **Example 3** shows a transaction dump table entry for transaction dump code AKC3. This is a CICS transaction abend, and this dump table entry is referenced every time a transaction abends AKC3—that is, whenever the master terminal operator purges a task.

  The entry shows that no action at all is to be taken in the event of such an abend. The maximum number of times the dump code action can be taken is given as 999, meaning that there is no limit to the number of times the specified action is taken. The dump code action has been taken 37 times, but each time both the transaction dump and the system dump were suppressed.

Table 35 on page 232 shows how the transaction dump table entry for transaction dump code MYAB would be updated with and without global suppression of system dumping. Only the updated fields are shown.

Table 35. Effect of global suppression of system dumping on transaction dump table update			
Type of information	**Before update**	**System dumping enabled**	**System dumping suppressed**
Transaction dump code	MYAB		
Take a transaction dump?	YES		
Take a system dump?	YES		
Shut down CICS?	NO		
Maximum times action can be taken	50		
Times action already taken	0	1	1
Transaction dumps taken	0	1	1
Transaction dumps suppressed	0	0	0
System dumps taken	0	1	0
System dumps suppressed	0	0	1

The statistics show that a system dump was taken when system dumping was enabled, but not when system dumping was suppressed.

There is a further effect. CICS maintains a record of the **current dump ID**—the number of the most recent dump to be taken. This is printed at the start of the dump, together with the appropriate dump code. It is concatenated with the CICS run number—that is, the number of times that CICS has been brought up since the global catalog was created—to provide a unique ID for the dump.

Note: This does not apply to SDUMPs taken by the kernel; these always have a dump ID of 0/0000.

For example, for the ninth dump to be taken during the eleventh run of CICS, if the dump code were TD01, this is what you would see:

```
CODE=TD01 ID=11/0009
```

If system dumping is enabled for the dump code, the same dump ID is given to both the transaction dump and the system dump.

The system dump table

Table 36 shows two examples of the type of information that might be maintained in the system dump table for different system dump codes.

Table 36 (Page 1 of 2). Examples of system dump table entries		
Type of information	**Example 1**	**Example 2**
System dump code	SYDMP001	MT0001
Take a system dump?	YES	YES
Shut down CICS?	YES	NO
Maximum times action can be taken	(default)	999
Times action already taken	0	79

Table 36 (Page 2 of 2). Examples of system dump table entries		
Type of information	**Example 1**	**Example 2**
System dumps taken	0	79
System dumps suppressed	0	0

The type of information kept in the system dump table is similar to that kept in the transaction dump table (see Table 34 on page 230).

- **Example 1** shows a system dump table entry for system dump code SYDMP001, a user-supplied system dump code, specified using EXEC CICS PERFORM DUMP. The table entry shows that no dumps have yet been taken. However, if one were taken, CICS would be shut down. If global suppression of system dumping was in effect, no dump would be taken but CICS would be shut down if this dump code were referenced.
- **Example 2** shows the system dump table entry for system dump code MT0001, the CICS-supplied dump code for system dumps requested from the master terminal, with CEMT PERFORM DUMP or CEMT PERFORM SNAP. CICS is not shut down when a dump is taken for this dump code. Also, the value of 999 for "maximum times action can be taken" shows that an unlimited number of dumps can be taken for this dump code. The current count ("times action already taken") shows that to date, 79 dumps have been requested using CEMT.

What happens to a dump request if there is no dump table entry?

If a dump is requested, either by CICS or the user, using a dump code that is not in the dump table, CICS makes a temporary dump table entry using default values for the attributes. However, the entry is not written to the CICS global catalog, and it is lost when CICS is shut down.

The **default** value for the maximum number of times that the dump action can be taken is set by the TRDUMAX system initialization parameter (for new or added transaction dump codes) and the SYDUMAX system initialization parameter (for new or added system dump codes).

You can modify the default values for a transaction dump table entry using the following commands:

- CEMT SET TRDUMPCODE
- EXEC CICS SET TRANDUMPCODE
- EXEC CICS SET TRANSACTION DUMPING (to modify the TRANDUMPING attribute only).

The following table shows the default values for transaction dump table entries and the attributes you can specify to modify them:

Table 37 (Page 1 of 2). Default values for transaction dump table entries			
Action	**Default**	**Attribute**	**Permitted value**
Take a transaction dump?	YES	TRANDUMPING	TRANDUMP or NOTRANDUMP

Table 37 (Page 2 of 2). Default values for transaction dump table entries			
Action	**Default**	**Attribute**	**Permitted value**
Take a system dump?	NO	SYSDUMPING	SYSDUMP or NOSYSDUMP
Shut down CICS?	NO	SHUTOPTION	SHUTDOWN or NOSHUTDOWN
Maximum times dump code action can be taken	999	MAXIMUM	0 through 999

You can modify the default values for a system dump table entry using the following commands:

- CEMT SET SYDUMPCODE
- EXEC CICS SET SYSDUMPCODE
- EXEC CICS SET SYSTEM DUMPING (to modify the SYSDUMPING attribute only).

The following table shows the default values for system dump table entries and the attributes you can specify to modify them:

Table 38. Default values for system dump table entries			
Action	**Default**	**Attribute**	**Permitted value**
Take a system dump?	YES	SYSDUMPING	SYSDUMP or NOSYSDUMP
Shut down CICS?	NO	SHUTOPTION	SHUTDOWN or NOSHUTDOWN
Maximum times dump code action can be taken	999	MAXIMUM	0 through 999

For example, if you issue a command requesting a dump, using the previously undefined dump code SYDMPX01:

```
EXEC CICS PERFORM DUMP DUMPCODE('SYDMPX01')
```

CICS makes a temporary dump table entry for dump code SYDMPX01 which you can browse, and see that it has the default attributes for a system dump code. You can also see that the current count has been set to 1, as a dump has been taken.

Attempting to add the dump code to the dump table after CICS has made the entry causes the exception response 'DUPREC' to be returned. If you want to make a change to the CICS-generated default table entry, and have that entry preserved across CICS runs, you must delete it and then add a new entry with the options you require.

Specifying the areas you want written to a transaction dump

When you use the EXEC CICS DUMP TRANSACTION command to get a transaction dump, you can specify which areas of storage are to be dumped. You cannot specify in the dump table which areas are to be written to the transaction dump for particular transaction dump codes. You always get a complete transaction dump whenever a transaction abend occurs, if the dump code requires a transaction dump to be taken.

The CSFE ZCQTRACE facility

The CSFE ZCQTRACE facility is used to gather information during the build process of terminal or connection definition. The syntax is as follows:

```
CSFE {ZCQTRACE=termid|ZCQTRACE,AUTOINSTALL|ZCQTRACE,OFF}
```

When CSFE ZCQTRACE is enabled, a dump of the builder parameter set and the appropriate TCTTE is written to the transaction dump data set at specific points in the processing. Table 39 shows the circumstances in which dumps are invoked, the modules that invoke them, and the corresponding dump codes.

Table 39. ZCQTRACE dump codes

Module	Dump code	When invoked
DFHTRZCP	AZQC	Installing terminal when termid = terminal ID
DFHTRZZP	AZQZ	Merging terminal with TYPETERM when termid = terminal ID
DFHTRZXP	AZQX	Installing connection when termid = connection ID
DFHTRZIP	AZQI	Installing sessions when termid = connection ID
DFHTRZPP	AZQP	When termid = pool terminal ID
DFHZCQIQ	AZQQ	Inquiring on resource when termid = resource ID (resource = terminal or connection)
DFHZCQIS	AZQS	Installing a resource when termid = resource ID (resource = terminal or connection), or when ZCQTRACE,AUTOINSTALL is specified.
DFHTRZYP	AZQY	Installing typeterm when ZCQTRACE,AUTOINSTALL is specified.

If a terminal definition is shipped from a terminal owning region (TOR) to an application owning region (AOR) and ZCQTRACE is enabled for that terminal in the TOR, then DFHZCQIQ invokes a dump in the TOR, and DFHZCQIS invokes a dump in the AOR.

Where dumps are written

Transaction dumps and system dumps are written to different destinations.

- **Transaction dumps** go to a pair of CICS SAM data sets, with DLBL names DFHDMPA and DFHDMPB. Some of the attributes of these data sets can be set by system initialization parameters, some can be set dynamically using CEMT SET DUMPDS or EXEC CICS SET DUMPDS, and all can be inquired

on by using CEMT INQUIRE DUMPDS or EXEC CICS INQUIRE DUMPDS. DFHDMPA and DFHDMPB have the following attributes:

- CURRENTDDS status. This tells you the data set that is currently active, which is the one where transaction dumps are currently written.

  You can use the system initialization parameter DUMPDS to specify the transaction dump data set that is to be opened during initialization. You can use the CEMT SET DUMPDS transaction or an EXEC CICS SET command to switch the dump data sets.

- One of the statuses OPEN or CLOSED. A transaction dump data set must be OPEN if it is to be written to.

- The current data set has one of the statuses AUTOSWITCH or NOAUTOSWITCH. You can set the status during system initialization using the DUMPSW system initialization parameter, and you can set it dynamically using the CEMT transaction or an EXEC CICS SET DUMPDS command.

  If the status is AUTOSWITCH, a switch is made automatically to the other dump data set when the current one becomes full, and subsequent transaction dumps are written to the new dump data set. The entire overflowing transaction dump is written in to the new dump data set.

  The dump data set being switched to does not inherit the AUTOSWITCH status, to prevent data in the first dump data set from being overwritten by another switch. You need to reset the AUTOSWITCH status explicitly, if you want it.

  If the status is NOAUTOSWITCH, a switch is not made when the current dump data set becomes full, so no more transaction dumps can be written.

- **CICS system dumps** are written to a VSE SYSDUMP library, by an invocation of the VSE SDUMPX macro.

Analyzing dumps

CICS system dumps and transaction dumps are unformatted when written to the appropriate dump data set. In other words, they are memory dumps of all or part of the CICS address space.

Unformatted dumps are not easy to interpret, and you are recommended not to use them for debugging. CICS provides utilities for formatting transaction dumps and CICS system dumps, and you should always use them before you attempt to read any dump. You can quickly locate areas of storage that interest you in a formatted dump, either by browsing it online, or by printing it and looking at the hard copy.

The formatting options that are available for transaction dumps and system dumps are described in “Formatting transaction dumps” and “Formatting system dumps” on page 244, respectively.

Formatting transaction dumps

You can format transaction dumps offline using the CICS dump utility program, DFHDU410. Individual transaction dumps must be formatted in their entirety, but you can control which dumps are formatted from the dump data set. You can select dumps to be formatted as follows:

- Those taken in a specific period of time
- Those associated with a specific transaction identifier
- Those taken for a specific transaction dump code
- Those having specific dump IDs—that is, those for specific CICS runs and dump count values

You can also use the SCAN option with the dump utility program, to get a list of the transaction dumps recorded on the specified dump data set.

For information about using DFHDU410 to format transaction dumps, see the *CICS Operations and Utilities Guide*.

Interpreting transaction dumps

This section describes the contents of a transaction dump, and gives guidance on locating information that is useful for debugging transactions. The different parts of the transaction dump are dealt with in the order in which they appear, but be aware that only those parts that will help you for problem determination are described. Some control blocks which do appear in the transaction dump are intended for the problem determination purposes of IBM Service and, as such, are not described in this section.

Job log

The job log for the dump utility program, DFHDU410, is sometimes shown at the start of the transaction dump, depending on how the job was invoked. You can ignore it, because it does not contain any information that can be used for debugging.

Symptom string

The symptom string tells you something about the circumstances of the transaction dump. It might show, for example, that the dump was taken because the transaction abended with abend code ASRA.

If you refer the problem that caused the dump to be taken to the IBM Support Center, they can use the symptom string to search the RETAIN database for problems that resemble it.

CICS Transaction Server for VSE/ESA level

The CICS Transaction Server for VSE/ESA level that was being executed when the transaction dump was taken.

Transaction environment summary

The transaction environment summary is a formatted summary of the transaction environment at the time the dump is taken.

Remote abend indicator

If the transaction abended remotely (the abend originally occurred in a remote distributed program link (DPL) server program), and the abend is being re-issued on the local system, a message indicates this. The message contains the SYSID of the system that passed the abend to the local system. This information is taken from the transaction abend control block (see “Transaction storage” on page 240).

PSW

If the transaction dump was taken in response to a local abend with abend code AICA, ASRA, ASRB, or ASRD, a PSW is formatted from the dump data set. It belongs to the program that was being executed when the abend occurred. It is taken from the transaction abend control block (see “Transaction storage” on page 240).

Registers at the time of interrupt

If the transaction abended locally with abend code AICA, ASRA, ASRB, or ASRD, you see a set of registers that belong to the program that was executing when the error was detected. They are taken from the transaction abend control block (see “Transaction storage” on page 240).

Execution key

If the transaction abended locally with abend code ASRA or ASRB, the execution key that is in force at the time of the abend is formatted. It is taken from the transaction abend control block (see “Transaction storage” on page 240).

Registers at last EXEC command

If the transaction has issued any EXEC commands, then a set of registers is displayed. These are the registers from the last EXEC command that was issued.

Task control area (TCA)

The next thing in the transaction dump is the entire TCA. This contains information about the transaction to which the dump relates. Note that the user area precedes the system area.

Task control area, system area

The system area of the task control area is formatted separately as it can be addressed separately by CICS. It contains system information relating to the task and can often be a valuable source of debugging information.

Transaction work area (TWA)

Any transaction work area relating to the transaction is formatted, if present.

EXEC interface structure (EIS)

The EIS contains information about the transaction and program specific to the execution interface component of CICS.

System EXEC interface block (SYSEIB)

This is used solely by programs using the SYSEIB option. See the *CICS Application Programming Guide* for more details.

EXEC interface user structure (EIUS)
The EIUS contains execution interface component information that must reside in user key storage.

EXEC interface block (DFHEIB)
DFHEIB contains information relating to the passing of EXEC requests from the program to CICS, and the passing of data between the program and CICS. Field EIBFN is of particular interest, because it shows the type of the last EXEC command to be issued by the program. For programming information about the values that EIBFN can contain, see the *CICS Application Programming Reference* manual or the *CICS User's Handbook*.

Kernel stack entries
The kernel stack entries contain information that has been saved by the kernel on behalf of programs and subroutines on the kernel linkage stack. If you refer the problem to the IBM Support Center, they might need to use the stack entries to find the cause of the failure.

Common system area (CSA)
The CSA is one of the main control areas used by CICS. It contains information relating to the system as a whole, and to the task that was running when the transaction dump was invoked. It can be very useful for debugging both application problems and system problems. You cannot access fields in the CSA in your programs. Attempting to do so causes your transaction to terminate abnormally with abend code ASRD.

Common system area optional features list (CSAOPFL)
The common system area optional features list, an extension of the CSA, contains the addresses of CICS optional features.

Trace table (abbreviated format)
The abbreviated-format trace table is formatted by default. You can suppress it by specifying the NOABBREV parameter in the DFHDU410 job. A "one entry per line" summary of the trace entries, copied from the internal trace table, is formatted.

Provided that you had EI level-1 and PC level-1 tracing selected for your task, you can identify the last CICS command issued by your task quite easily. The procedure is outlined in "Locating the last command or statement" on page 241.

Trace table (extended format)
Following the abbreviated-format trace table is the corresponding extended-format trace table. This is formatted by default. You can suppress it by specifying the NOFULL parameter in the DFHDU410 job. It contains more detail, but you can probably get all the information you need using the abbreviated trace.

Common work area (CWA)
The CWA is the installation-defined work area for use by all programs and is formatted if it exists.

Transaction storage

"Transaction storage" is storage that might have been obtained by CICS to store information about a transaction, or it might have been explicitly GETMAINed by the transaction for its own purposes. You are likely to find several such areas in the dump, each introduced by a header describing it as transaction storage of a particular class, for example:

USER24
USER31
CICS24
CICS31

Transaction storage class CICS31 contains, among other things, the transaction abend control block (TACB). To find it, look for the eye-catcher **DFHTACB**. DFHTACB contains valuable information relating to the abend. It contains:

- The PSW and general purpose registers of the program executing at the time of the abend (for local AICA, ASRA, ASRB, and ASRD abends only. However, for some AICA abends, only the "next sequential instruction" part of the PSW and the registers are present.)
- The name of the failing program.
- The offset within the failing program at which the abend occurred (for local ASRA, ASRB and ASRD abends only).
- The execution key at the time of the abend (for local ASRA and ASRB abends only).
- Whether the abend was caused by an attempt to overwrite a protected CICS DSA (local ASRA abends only).

Note that if the abend originally occurred in a remote DPL server program, an eye-catcher ***REMOTE*** is present. If this is the case, the registers and PSW are not present.

Terminal control table terminal entry (TCTTE)

The TCTTE contains information about the terminal that is the principal facility of the transaction. You usually find one TCTTE for the transaction if the transaction is terminal oriented. For "daisy chained" transactions, you may find more than one.

To look at the TIOA for the task, find the address in field TCTTEDA of the TCTTE.

Program information for the current transaction

This shows summary information for all linked programs for the transaction that have not yet returned, including load point, entry point, and length. This is followed by the program storage for each of these programs. This is where you can find the instructions addressed by register 14 and by the PSW, and hence the point of failure in your program. For details of how you do this, see "Locating the last command or statement" on page 241.

Other program manager control blocks are shown too, including PPT entries for active programs, and load list elements and program storage for any programs loaded by this transaction but not yet released.

Module index

The final item that you find in the transaction dump is the module index. This shows you all the modules that were in storage when the error was detected, and their addresses.

Locating the last command or statement

The easiest way of locating the last command issued by your application before the abend is to use the internal trace table. You must have had internal trace running, and you need to have captured entries from the EI level-1 and PC level-1 trace points for your task. The way you can find the last command using trace is described in "Last command identification."

If you did not have trace running, you need to rely on values in registers belonging to your programs, and try to relate these to your source statements. That might lead you to an EXEC CICS command, or to another type of statement in your program. The procedure is outlined in "Last statement identification" on page 242.

Last command identification

This process for identifying the last command applies to system dumps, in which it is necessary to identify the abending task. Item 2 is relevant to transaction dumps.

1. If the abend was a local AICA, ASRA, ASRB, or ASRD, find the last entry in the table and work back until you find an entry for trace point ID AP 1942. If you cannot find AP 1942, then search for any one of the following: AP 0790, AP 0791, or AP 0792. The trace entry for AP 1942 is made on entry to APLI's recovery routine. The entries for AP 0790, AP 0791, and AP 0792 are made by DFHSRP, the AP domain recovery routine that deals with program checks, operating system abends, and runaway task conditions. The task number for your task is shown in the entry.

   If the abend was none of those mentioned above, find the last entry in the table and work back until you find an entry for trace point ID AP 00F2 (PCP abend) that references the abend code. The task number of your task is shown in the entry.

2. Now go back until you find the last trace entry showing your task number that was made from trace point ID AP 00E1. The trace entry is located in the EXEC interface program, DFHEIP. The data in the trace entry includes the value of EIBFN, which tells you the specific command your program issued before the abend. For programming information about the possible values that EIBFN can take, and their meanings, see the *CICS Application Programming Reference* manual, the *CICS System Programming Reference* manual or the *CICS User's Handbook*.

3. You might now be able to identify the program that was being run when the abend occurred, from knowing the structure of the application. If not, you can identify the program by using the information in the "program information for the current transaction" section of the dump. The failing program is the one most recently linked to (the first program printed in this section). The summary information includes the name of the program, and its load point, entry point, and length.

4. You should by now have found the program containing the last EXEC command that was issued before the abend. Now locate the EXEC command

in that program. If you cannot do it by inspection, use the techniques described in the next section.

Last statement identification

1. Locate the “transaction storage –CICS31” areas of the transaction dump. These areas are maintained by CICS, and they relate to the transaction that was running when the abend occurred. You should be able to see the eye-catcher **DFHTACB** in at least one of the areas. This signifies the start of the transaction abend control block, and it contains the registers and PSW belonging to the program being executed when the abend occurred. If there is more than one area containing this eye-catcher, it means that two or more successive abends occurred. Find the first occurrence, because that relates to the abend that started the sequence.
2. Locate the PSW for the program in the TACB, and make a note of the next sequential instruction address. The PSW for the program is present if the abend is AICA, ASRA, ASRB or ASRD. Alternatively, obtain the offset of the abend within the failing program load module from the TACB. The offset is present if the abend is ASRA, ASRB or ASRD and is valid if not `X'FFFFFFFF'`. Note down the value of register 14, too.
3. Use the “program information for the current transaction” section of the dump to obtain the name and entry point of the failing program. Alternatively, obtain the name of the failing program from the TACB.
4. The offset or PSW should point to an instruction in the program. If it does not, register 14 should show a return address into your program. Either way, you need to correlate the address with a statement in the source code for the program.

   Note: In system dumps formatted by DFHPD410, the offset of an error is formatted in the kernel error data section. Because the EXEC interface stub is linked at the start of the program code, you need to subtract `X'28'` bytes from this offset in order to calculate the actual offset of the failing instruction.

   If the source language is **assembler**, the instruction where the abend occurred is apparent from the program storage in the dump. If the source language is **COBOL**, **PL/I**, or **C**, you need to refer to a compiler output mapping source statements onto the object code.

Locating program data

You can look at the data that your application program has in its storage areas. CICS maintains a pointer to the chain of dynamic storage that the program uses, in field TCAPCDSA of the system area of the TCA.

- For **PL/I programs**, TCAPCDSA addresses the chain of PL/I DSAs.
- For **COBOL programs**, TCAPCDSA addresses the task global table (TGT) and working storage.
- For **assembler programs**, TCAPCDSA addresses the DFHEISTG storage.

Look in the appropriate programming language manual for details of the structure of the program’s acquired storage.

Dumps for C programs

If DUMP(YES) is coded on the RDO TRANSACTION definition, CICS system and transaction dumps are produced for failing C programs. The use of the relevant C registers is as follows:

Register	Use
3	In most circumstances, is the base register
12	Holds the address of the CICS TCA for the C program
13	Holds the address of the register save area

Location of COBOL dumped areas

You can find the dumped COBOL program in the “program information for the current transaction” section of the dump—it is addressed by the LOAD_POINT parameter on the appropriate LDLD ACQUIRE_PROGRAM exit trace entry. The register save area INIT1+X'48' (covering registers 0 through 14) should have register 12 pointing to the program global table (PGT), register 13 pointing to the task global table (TGT), and some others to locations in the data area and compiled code of the program storage. If not, a CICS error is indicated.

For each invocation of the COBOL program, CICS copies the **static** TGT from program storage into CICS **dynamic** storage (the COBOL area) and uses the dynamic copy instead of the static one. CICS also copies working storage from program storage to the COBOL area, above the TGT. Task-related COBOL areas thus enable the single copy of a COBOL program to multithread as if it were truly reentrant.

The dumped COBOL area

The COBOL area is addressed by TCAPCDSA (alias TCAPCCA) in the system part of the TCA (and forms part of the transaction storage chain). The COBOL area contains:

- The COBOL working storage for the task
- A copy of the TGT

At any time after the program has issued a CICS request, the TGT is always addressed by TCAPCHS in the system part of the TCA.

The TGT is used to hold intermediate values set at various stages during program execution. The first 18 words of the TGT constitute a standard save area, in which the program’s current registers are stored on any request for CICS service.

The TGT also holds the base locator for linkage (BLL) cells, which contain the addresses of all areas defined in the linkage section of the COBOL program. You can use the memory map from the COBOL compiler listing for your application to find the offset of the BLL cells in the TGT.

BLL cells in a CICS application program

To investigate a problem connected with BLL cells, you should examine the COBOL compiler listing for your application to determine the use you have made of the BLL cells.

The **first** BLL cell that is used points to the EXEC interface block (an area that follows the literal string 'DFHEIB').

The **second** BLL cell that is used points to the first (or only) 4096-byte block of CICS COMMAREA as supplied with an EXEC CICS LINK, XCTL, or RETURN with a TRANSID. If the COMMAREA is greater than 4096 bytes, the **third** BLL cell that is used points to the next 4096-byte block of COMMAREA; further 4096-byte COMMAREAs are pointed to by succeeding BLL cells.

The next BLL cell always points to itself; the remaining BLL cells are set by the application program. Thus, for example, if the COMMAREA size is 10KB, it is the **fifth** BLL cell used that points to itself.

Note that an ASRD abend will occur if an attempt is made to access storage via the BLL cell immediately after the one that points to itself, if it has not yet been set by the application program.

Formatting system dumps

You can process VSE system dumps (SDUMPs) from the VSE dump library (SYSDUMP) using the VSE Info/Analysis program (INFOANA).

In releases prior to CICS Transaction Server for VSE/ESA Release 1, the CICS system dump formatting routine for use with INFOANA is supplied as DFHDAP. This standard name is not suitable for those users running more than one release of CICS, because the dump formatting processing in each version is release-specific, and you must use the correct version for the system dump you are formatting. The module is now named with the release identifier as part of the name—DFHPD410 is the formatting routine you must use with INFOANA when formatting CICS Transaction Server for VSE/ESA Release 1 system dumps.

The CICS SDUMP formatting routine (DFHPD410) enables you to format a system dump selectively by specifying one or more CICS component identifiers as parameters to the formatting routine. Thus the CICS SDUMP formatting routine enables you to process a CICS Transaction Server for VSE/ESA Release 1 system dump selectively. You can:

- Specify which component storage areas are to be formatted,and at what level of detail, by using formatting keywords and level numbers (KEYWORD parameter)
- Specify the default level of detail for components that are not explicitly identified by keyword (DEF parameter)
- Specify whether the output is to be printed in uppercase characters (UPPERCASE parameter)

A summary of system dump formatting keywords and levels

The component keywords specify which areas of CICS you want the INFOANA exit to format dump data for, and the level number operand specifies the amount of data you want formatted.

If you omit all of the component keywords, and you do not specify DEF=0, the CICS dump exit formats dump data for all components.

The CICS dump component keywords, and the levels you can specify for each of them, are as follows:

AI [={0|2}]
Autoinstall model manager.

- **AI=0** Suppress formatting of AI control blocks.
- **AI=2** Format AI control blocks.

AP [={0|1|2|3}]
Application domain.

- **AP=0** Suppress formatting of AP control blocks.
- **AP=1** Format a summary of addresses of the AP control blocks for each active transaction.
- **AP=2** Format the contents of the AP control blocks for each active transaction.
- **AP=3** Format level-1 and level-2 data.

CC [={0|2}]
The CICS catalog domain.

- **CC=0** Suppress formatting of CC control blocks.
- **CC=2** Format the CC control blocks.

CP [={0|2}]
The common programming interface.

- **CP=0** Suppress formatting of CP control blocks.
- **CP=2** Format the CPI static storage.

CSA[={0|2}]
The CICS common system area.

- **CSA=0** Suppress formatting of the CSA.
- **CSA=2** Format the CSA and its extension, the optional features list (CSAOPFL).

DD[={0|1|2|3}]
The directory manager domain.

- **DD=0** Suppress formatting of DD control blocks.
- **DD=1** Format the directory manager summary.
- **DD=2** Format the directory manager control blocks, including the anchor block, directory headers, and AVL tree headers.
- **DD=3** Format level-1 and level-2 data.

DM[={0|1|2|3}]
The domain manager.

DM=0 Suppress formatting of DM control blocks.

DM=1 Format the wait queue.

DM=2 Format the anchor block.

DM=3 Format level-1 and level-2 data.

DS[={0|1|2|3}]
The dispatcher domain.

DS=0 Suppress formatting of DS control blocks.

DS=1 Format the dispatcher dump summary.

DS=2 Format the anchor block.

DS=3 Format level-1 and level-2 data.

DU[={0|2}]
The dump domain.

DU=0 Suppress formatting of DU control blocks.

DU=2 Format the DU anchor block.

FCP[={0|2}]
The file control program.

FCP=0 Suppress formatting of the file control table.

FCP=2 Format the file control table.

ICP[={0|2}]
The interval control program.

ICP=0 Suppress formatting of the interval control elements (ICEs).

ICP=2 Format the ICEs.

IND[={0|1|2|3}]
The page number indexes for the formatted output.

IND=0 Suppress formatting of the page number indexes.

IND=1 Provide a control block index sorted by address.

IND=2 Provide a control block index sorted by block name.

IND=3 Format level-1 and level-2 data.

JCP [={0|2}]
The journal control area.

JCP=0 Suppress formatting of the JCA.

JCP=2 Format the JCA.

KE[={0|1|2|3}]
The CICS kernel.

KE=0 Suppress formatting of the kernel control blocks.

KE=1 Format the stack and a summary of tasks.

KE=2 Format the anchor block.

KE=3 Format level-1 and level-2 data.

LD[={0|1|2|3}]
The loader domain.

LD=0 Suppress formatting of loader domain control blocks.

LD=1 Format a program status and module summary.

LD=2 Format the anchor block, the current program elements (CPEs), and the active program elements (APEs)

LD=3 Format level-1 and level-2 data.

LM[={0|1|2|3}]
The lock manager domain.

LM=0 Suppress formatting of lock manager domain control blocks.

LM=1 Format the lock status and allocated locks summary.

LM=2 Format the anchor block and quickcells.

LM=3 Format level-1 and level-2 data.

ME[={0|2}]
The message domain.

ME=0 Suppress formatting of the ME anchor block.

ME=2 Format the anchor block.

MN[={0|1|2|3}]
The monitoring domain.

MN=0 Suppress formatting of monitoring domain control blocks.

MN=1 Format the monitoring status and monitoring dictionary summary.

MN=2 Format the anchor block and monitoring control table.

MN=3 Format level-1 and level-2 data.

MRO[={0|1|2|3}]
CICS multiregion operation. APPC URDs, and any associated DWE chains are formatted on all non-zero levels.

MRO=0 Suppress formatting of all MRO control blocks.

MRO=1 Format MRO control blocks for the current address space.

MRO=2 Format MRO control blocks for all address spaces except the current address space.

MRO=3 Format level-1 and level-2 data.

PA[={0|2}]
The parameter manager domain.

PA=0 Suppress formatting of the PA anchor block.

PA=2 Format the PA anchor block.

PCT[={0|2}]
The program control table.

PCT=0 Suppress formatting of the transaction definitions.

PCT=2 Format the transaction definitions.

PG[={0|1|2|3}]
The program manager domain.

PG=0 Suppress formatting of program manager domain control blocks.

PG=1 Format the program manager summary.

PG=2 Format the program manager control blocks, including the anchor block, the LLEs, the PGWEs, the PPTEs, the PLCBs, and the HTBs.

PG=3 Format level-1 and level-2 data.

PR [={0|2}]
Partner resource management.

PR=0 Suppress formatting of PR areas.

PR=2 Format the PR static storage and the partner resource table.

RC [={0|2}]
The report controller

RM=0 Suppress formatting of RC control blocks.

RM=2 Format RC control blocks.

RM [={0|2}]
The recovery manager domain.

RM=0 Suppress formatting of RM control blocks.

RM=2 Format RM control blocks.

SM[={0|1|2|3}]
The storage manager domain.

SM=0 Suppress formatting of storage manager domain control blocks.

SM=1 Format the dynamic storage areas (DSAs), task and domain storage subpools, transaction areas (SMXs), suspend queue, and subspace area (SUA) summaries.

SM=2 Format the anchor block (SMA), subpool control areas (SCAs), pagepool areas (PPAs), pagepool extent areas (PPXs), storage manager transaction areas (SMXs), subspace areas (SUAs), suspend queue elements (SQEs), page allocation maps (PAMs), DSA extent descriptors (DXEs), and DSA extent getmain descriptors (DXGs).

SM=3 Format level-1 and level-2 data.

SSA[={0|2}]
The static storage areas.

SSA=0 Suppress formatting of the static storage areas address list.

SSA=2 Format the static storage areas address list.

ST[={0|1|2|3}]
The statistics domain.

ST=0 Suppress formatting of statistics domain control blocks.

ST=1 Format statistics collection details.

ST=2 Format the anchor block.

ST=3 Format level-1 and level-2 data.

SZ[={0|1}]
Front end programming interface (FEPI).

SZ=0 Suppress formatting of the FEPI static area.

SZ=1 Format the FEPI static area. Nothing is printed unless you have installed FEPI. See the *CICS Front End Programming Interface User's Guide* for further information on this feature.

TCP[={0|1|2|3}]
The terminal control program.

TCP=0 Suppress formatting of TCP control blocks.

TCP=1 Format the terminal control summary.

TCP=2 Format the terminal control table, the terminal input/output areas, and the AIDs.

TCP=3 Format level-1 and level-2 data.

TDP[={0|1|2|3}]
The transient data program.

TDP=0 Suppress formatting of transient data control blocks.

TDP=1 Format the summary of transient data destinations.

TDP=2 Format the transient data static storage areas and the multiple strings control blocks (MRCBs).

TDP=3 Format level-1 and level-2 data.

TI[={0|1|2|3}]
The timer domain.

TI=0 Suppress formatting of timer domain control blocks.

TI=1 Format outstanding request details.

TI=2 Format the anchor block.

TI=3 Format level-1 and level-2 data.

TMP[={0|2}]
The table manager program.

TMP=0 Suppress formatting of table manager static storage and control blocks.

TMP=2 Format table manager static storage and control blocks.

TR[={0|1|2|3}
The trace domain.

TR=0 Suppress formatting of trace.

TR=1 Format abbreviated trace.

TR=2 Format full trace.

TR=3 Format level-1 and level-2 data.

TRS[={<trace selectivity parameter(s)>}]
Trace entry selectivity.

This keyword is effective only if the TR keyword value is 1, 2, or 3.

The TRS component keyword allows you to exercise much the same selectivity over the formatting and printing of trace entries written in a trace internal to a system dump, as you may exercise over the formatting and printing of trace entries in an auxiliary trace.

The trace selectivity parameter may be any valid trace selectivity parameter available to DFHTU410 for the formatting of CICS auxiliary trace entries except the parameters PAGESIZE, ABBREV, and FULL. You may, as with DFHTU410, select any number of parameters from those available. See the *CICS Operations and Utilities Guide* for more information about DFHTU410 trace selectivity parameters.

Note, however, that you must use angled brackets around the parameter, or sequence of parameters, that you specify. The format and default values of parameters used to select trace entries from an internal SDUMP trace, are the same as those that apply when you use DFHTU410 to format auxiliary trace entries.

TSP[={0|1|2|3}]
Temporary storage domain.

TSP=0 Suppress formatting of temporary storage control blocks.

TSP=1 Format a summary of temporary storage control blocks and temporary storage control block checking.

TSP=2 Format temporary storage control blocks.

TSP=3 Format level-1 and level-2 data.

UEH[={0|2}]
The user exit handler.

UEH=0 Suppress formatting of control blocks.

UEH=2 Format control blocks.

US[={0|1|2|3}]
The user domain.

US=0 Suppress formatting of user domain control blocks.

US=1 Format the user domain summary.

US=2 Format the control blocks.

US=3 Format level-1 and level-2 data.

XM[={0|1|2|3}]
The transaction manager.

XM=0 Suppress formatting of transaction manager control blocks.

XM=1 Format the domain summary, global state summary, transaction summary, transaction class summary, and MXT summary.

XM=2 Format the control blocks including the transaction domain anchor block, transactions (TXn), and transaction class control blocks (TCL).

XM=3 Format level-1 and level-2 data.

XRF[={0|2}]
The extended recovery facility.

XRF=0 Suppress formatting of control blocks.

XRF=2 Format control blocks.

XS[={0|1}]
The security domain.

XS=0 Suppress formatting of anchor block and supervisor storage.

XS=1 Format anchor block and supervisor storage.

For guidance about using the formatting keywords and levels, see the *CICS Operations and Utilities Guide*.

The default SDUMP formatting levels

The **DEF** dump exit parameter specifies the default level of formatting you want for data from the dump data set. Note that the DEF parameter is only effective for components that are not included in a list of component keywords.

The levels that you can specify are as follows:

Level Meaning

0 For those components not included in a specified list of keywords, suppress all component formatting. Note that if you specify DEF=0, but do not specify any component keywords, you still get the dump summary and, if appropriate, the error message index.

1 For those components not included in a specified list of keywords, and where applicable, format the summary information only. (A summary is not available for all components; see the level numbers available for the individual keywords for those for which a summary of dump information is available.)

2 For those components not included in a specified list of keywords, format the control block information only.

3 For those components not included in a specified list of keywords, format the control block information and also (where applicable) the summary information.

Effects of omitting the DEF parameter:

- If you omit the DEF parameter and *do not* specify any component keywords, the result is as if you specified DEF=3.
- If you omit the DEF parameter *and* specify one or more component keywords, the result is as if you specified DEF=0.

Exceptions to the scope of the DEF parameter: There are two parts of a CICS system dump that are not governed by component keywords, and are therefore outside the scope of the DEF parameter. These are:

1. The dump summary
2. The error message index.

These parts of a CICS system dump are always formatted, even if you specify DEF=0 and no component keywords.

Storage freeze

Particular classes of CICS storage that are normally freed during the processing of a transaction can, optionally, be kept intact and freed only at the end of the transaction. Then, in the event of an abend, the dump contains a record of the storage that would otherwise have been lost, including the storage used by CICS service modules. The classes of storage that can be frozen in this way are those in the teleprocessing and task subpools, and in terminal-related storage (TIOAs).

The storage freeze function is invoked by the CSFE transaction. For information about using CSFE, see the *CICS-Supplied Transactions* manual.

Using FEPI dump

For information about using dumps to solve FEPI problems, see the *CICS Front End Programming Interface User's Guide*.

Chapter 16. The global trap/trace exit

The global trap/trace exit (DFHTRAP) is intended to be used only under the guidance of IBM Service personnel. It is designed so that a detailed diagnosis of a problem can be made without having to stop and then restart CICS.

Typically, the global trap/trace exit is used to detect errors that cannot be diagnosed by other methods. These might cause intermittent problems that are difficult to reproduce, the error perhaps occurring some time before the effects are noticed. For example, a field might be changed to a bad value, or some structure in storage might be overlaid at a specific offset.

DFHTRAP is an assembler language program that can be invoked whenever the trace (TR) domain is called to make a trace entry. The trap must be activated before it is used, either dynamically or at CICS initialization.

A skeleton version of DFHTRAP is supplied in both source and load-module forms. The source of DFHTRAP is cataloged in the VSE/ESA sublibrary PRD1.BASE. The source of the skeleton DFHTRAP contains comments explaining its use of registers and DSECTs, and the coding you need to do if you need to use the exit.

The code in DFHTRAP must **not** make use of any CICS services, cause the current task to lose control, or change the status of the CICS system.

Establishing the exit

DFHTRAP can be installed as part of the RDO group DFHFE. You can install it either by adding the DFHFE group to a group list specified on the GRPLIST system initialization parameter, or dynamically using the CEDA transaction.

When CICS is running, activation and deactivation of the trap exit routine can be requested using the TRAP operand of the CSFE DEBUG command:

```
CSFE DEBUG,TRAP={ON|OFF}
```

The trap exit can also be activated at CICS initialization by specifying TRAP=ON, either in DFHSIT or as a startup override. If you want to **replace** a trap exit routine while CICS is running, use the CSFE DEBUG commands in conjunction with the CEMT NEWCOPY command. The following sequence of commands causes the currently active version of DFHTRAP to be refreshed:

```
CSFE DEBUG,TRAP=OFF
CEMT SET PROGRAM(DFHTRAP) NEWCOPY
CSFE DEBUG,TRAP=ON
```

© Copyright IBM Corp. 1979, 1999

Information passed to the exit

To assist in the diagnosis of faults in a CICS system, the trace domain passes information to the exit in a parameter list addressed by register 1. A DSECT (DFHTRADS) is supplied for this list, which contains the addresses of:

- The return-action flag byte
- The trace entry that has just been added to the table
- Up to three areas to be included in a further trace entry
- An 80-byte work area for the sole use of the trap exit
- The CSA (which may be zero during early initialization)
- The TCA, if there is one
- A register save area

The DSECT also contains EQU statements for use in setting the return-action flag byte.

The exit can look at data from the current trace entry to determine whether or not the problem under investigation has appeared. It can also look at the TCA of the current task, and the CSA. The DSECTs for these areas are included in the skeleton source.

The CSA address is zero for invocations of DFHTRAP early in initialization, before the CSA is acquired.

Actions the exit can take

The return-action flag byte can be set by the exit to tell the trace domain what action is required on return from the exit. The following list gives the possible choices:

- Do nothing.
- Make a further trace entry.
- Take a CICS system dump using dump code TR1003.
- Terminate CICS without a dump after message DFHTR1000. (If you require a dump, the system dump return-action flag must also be set.)
- Disable the trap exit, so that it is not invoked again until reactivated by the CSFE transaction.

Any combination of these actions can be chosen and all actions are honored on return to the trace domain.

To reactivate the trap exit when it has been disabled, use CSFE DEBUG,TRAP=ON, unless the exit routine is to be replaced. In this case the sequence of commands given above applies.

The skeleton program shows how to make a further trace entry. When DFHTRAP detects a TS GET request, it asks for a further trace entry to be made by entering the data required in the area supplied for this purpose, and by setting the appropriate bit in the return-action flag byte.

The trace domain then makes a trace entry with trace point ID TR 0103, incorporating the information supplied by the exit.

Trace entries created in this way are written to any currently active trace destination. This could be the internal trace table or the auxiliary trace data set.

The skeleton DFHTRAP also shows how to detect the trace entry made by the storage manager (SM) domain for a GETMAIN request for a particular subpool. This is provided as an example of how to look at the data fields within the entry.

Program check handling

The occurrence of a program check in the trap exit is detected by the recovery routine in DFHTRPT. This then:

1. Marks the exit as unusable
2. Issues the message DFHTR1001 to the system console
3. Takes a CICS system dump with dump code TR1001, showing the PSW and registers at the time of the interrupt
4. Continues (ignoring the exit on future invocations of the trace domain).

To recover from this situation, execute the commands given above for replacing the current version of the exit routine.

Coding the exit

When using the trap exit, note the following points:

- A skeleton version of DFHTRAP is supplied in both source and load-module forms. Make sure that the LIBDEF search chain sequence in the CICS startup JCL finds the correct version of the phase.
- DFHTRAP must save and restore the trace domain's registers. The supplied skeleton version contains the code necessary to do this. You are strongly advised not to change this code.
- The 80-byte work area provided for the sole use of the exit is in working storage acquired by the trace domain using a VSE GETVIS. It is acquired and initialized to binary zeros when the trap is activated. It then exists until the trap is deactivated (CSFE DEBUG,TRAP=OFF). In a CICS transaction dump, the DFHTRAP working storage can be found soon after the CSA optional features list. The 80-byte work area is at the end of the DFHTRAP working storage and is immediately preceded by a 16-byte eye-catcher (**DFHTRAP_WORKAREA**), so that the work area can be located even if it has not been formatted. In a CICS system dump, the DFHTRAP working storage is in the trace domain (TR) section. See "Formatting system dumps" on page 244 for details of how to use the TR keyword to format the trace domain information in the dump.
- DFHTRAP must always run with AMODE(31) and RMODE(ANY) specified. In particular, it must always return control to the trace domain in 31-bit addressing mode.

Part 4. Working with IBM to solve your problem

Part 4 contains:

Chapter 17. IBM program support 259
When to contact the Support Center 259
Dealing with the Support Center 259
IBM Program Support structure 262
Reporting a FEPI problem to IBM 263

Chapter 18. APARs, fixes, and PTFs 265
The APAR process 265
Collecting the documentation for the APAR 265
Sending the documentation to the change team 266
Applying the fix 267

© Copyright IBM Corp. 1979, 1999

Chapter 17. IBM program support

The IBM Customer Engineering Program Support structure exists to help you resolve problems with IBM products, and to ensure that you can make the best use of your IBM computing systems. Program support is available to all licensed users of IBM licensed programs, and you can get assistance by telephoning your local Support Center.

This chapter helps you decide when to contact the Support Center, and what information you need to collect before contacting the Center. The chapter also gives you an understanding of the way in which IBM Program Support works.

When to contact the Support Center

Before contacting the Support Center, try to ensure that the problem belongs with the Center. Do not worry if you cannot be sure that the problem is due to CICS itself.

In practice, many errors reported to Program Support turn out to be user errors, cannot be reproduced, or need to be dealt with by other parts of IBM Service such as Hardware CE or Systems Engineering. This indicates just how difficult it can be to determine the precise cause of a problem. User errors are mainly caused by faults in application programs and errors in setting up systems. TCT parameters, in particular, have been found to cause difficulty in this respect.

Dealing with the Support Center

Your first contact with the Support Center is the call receipt operator, who takes initial details and routes your call to the correct support group.

The Support Center needs to know as much as possible about your problem, so have the information ready before making your first call. It is a good idea to record the information on a problem reporting sheet, such as the one shown in Figure 41 on page 260.

© Copyright IBM Corp. 1979, 1999

```
PROBLEM REPORTING SHEET

Date                  Severity              Problem  No.

                                            Incident No.

-------------------------------------------------------------

Problem/Enquiry

Abend/Prog CK         Incorrout             VSE Rel

Wait                  Module                VSE Lvl

Loop                  Message               CICS Rel

Performance           Other                 CICS Lvl

-------------------------------------------------------------

Documentation available

Abend                 System dump           Program output

Message               Transaction dump      Other

Trace                 Translator output

Symptom string        Compiler output

-------------------------------------------------------------

Actions

Date      Name        Activity

-------------------------------------------------------------

Resolution

APAR                  PTF                   Other
```

Figure 41. Sample problem reporting sheet

There are two advantages of using a problem reporting sheet:

1. You will be communicating with the IBM Support Center by telephone. So, with all your findings before you on a sheet of paper, you are prepared to respond to the questions that you may be asked.
2. You should maintain your own in-house tracking system to record all problems. This information can then be used for planning, organizing, communicating, and establishing priorities for controlling and solving these problems.

What the Support Center needs to know

When you contact the Support Center, you need to give the operator the name of your organization and your access code. Your access code is a unique code authorizing you to use IBM Software Services, and you provide it every time you contact the Center. Using this information, the operator accesses your customer profile, which contains details of your address, relevant contact names, telephone numbers, and details of the IBM products at your installation.

The Support Center operator asks you if this is a new problem, or a further call on an existing one. If it is new, you are assigned a unique incident number. A problem management record (PMR) is opened on the RETAIN system, where all activity associated with your problem is recorded. The problem remains “open” until it is solved.

Make a note of the incident number on your own problem reporting sheet. The Center expects you to quote the incident number in all future calls connected with this problem.

If the problem is new to you, the operator asks you for the source of the problem within your system software—that is, the program that seems to be the cause of the problem. As you are reading this book, it is likely that you have already identified CICS TS for VSE/ESA as the problem source. You also need to give the version and release number, for example Version 1 Release 1.

You need to give a severity level for the problem. Severity levels can be 1, 2, or 3. They have the following meanings:

- Severity level 1 indicates that you are unable to use a program, resulting in a critical condition that needs immediate attention.
- Severity level 2 indicates that you are able to use the program, but that operation is severely restricted.
- Severity level 3 indicates that you are able to use the program, with limited functions, but the problem is not critical to your overall operation.

When deciding the severity of the problem, take care neither to understate it nor to overstate it. The Support Center procedures depend on the severity level so that the most appropriate use can be made of the Center’s skills and resources. Your problem is normally dealt with immediately if it is severity level 1.

Finally, the operator offers you a selection of specific component areas within CICS (for example, terminal control, file control) and asks you to choose the area where your problem appears to lie. Based on this selection, the operator can route your call to a specialist in the chosen area.

The keywords are subsequently used as search arguments on the RETAIN database, to see if your problem is a known one that has already been the subject of an authorized program analysis report (APAR).

You are not asked for any more information at this stage. However, you need to keep all the information relevant to the problem, and any available documentation such as dumps, traces, and translator, compiler, and program output.

How your problem is subsequently progressed depends on its nature. The representative who handles the problem gives you guidance about what is required from you. The possibilities are described in the next section.

What happens next

Details of your call are passed using the RETAIN problem management system to the appropriate support group. Your problem, assuming it is one associated with CICS, is put on the CICS queue. The problems are dealt with in order of severity level.

At first, a support center representative uses the keywords that you have provided to search the RETAIN database. If your problem is found to be one already known to IBM, and a fix has been devised for it, a Program Temporary Fix (PTF) can quickly be dispatched to you.

Let the representative know if any of the following events occurred before the problem appeared:

- Changes in level of VSE or licensed programs
- Regeneration of any product
- PTFs applied
- Additional features used
- Application programs changed
- Unusual operator action

You might be asked to give values from a formatted dump or trace table. You might also be asked to carry out some special activity, for example to set a trap, or to use trace with a certain type of selectivity, and then to report on the results.

It might be necessary to have several follow-up telephone calls, depending on the complexity of the symptoms and your system environment. In every case, the actions taken by you and the Support Center are entered in the PMR. The representative can then be acquainted with the full history of the problem before any follow-up call.

The result of the investigations determines whether your problem is a new one, or one that is already known. If it is already known, and a fix has been developed, the fix is sent to you.

If the problem is new, an APAR may be submitted. This is dealt with by the CICS change team. See Chapter 18, “APARs, fixes, and PTFs” on page 265.

IBM Program Support structure

Figure 42 on page 263 shows the relationships between the customer, the staff at the IBM Support Center, and the change team. The role of each member of the IBM support staff is outlined in the rest of this section.

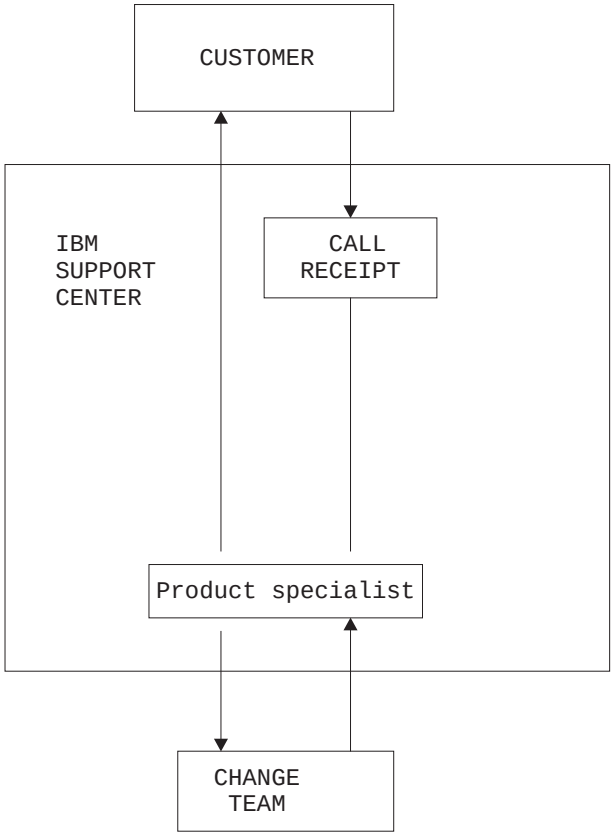


Figure 42. Structure of IBM Program Support

Reporting a FEPI problem to IBM

For information specifically about reporting a FEPI problem, see the *CICS Front End Programming Interface User's Guide*.

Chapter 18. APARs, fixes, and PTFs

An APAR is an “authorized program analysis report”. An APAR is your means of informing the appropriate change team of a problem you have found with an IBM program.

When the change team solves the problem, they produce a fix enabling you to get your system running properly again. Finally, a PTF is produced to replace the module in error, and the APAR is closed.

The APAR process

The first step in the APAR process is that a support center representative enters your APAR into the RETAIN system. The APAR text contains a description of your problem. If you have found a means of getting round the problem, details of this are entered as well. Your name is also entered, so that the Support Center knows who to contact if the change team need to ask anything further about the APAR documentation.

When the APAR is entered, you are given an APAR number. You must write this number on all the documentation you submit to the change team. This number is always associated with the APAR and its resolution and, if a code change is required, it is associated with the fix as well.

The next stage in the APAR process, getting relevant documentation to the change team, is up to you.

The following is a summary of the things you need to do:

1. You must collect all of the documentation that is required for the APAR. You are given guidance by the support center on precisely what you need to send. The documentation that is required varies, depending on the problem area, but “Collecting the documentation for the APAR” gives you an idea of the material that you should supply.
2. You need to package all the documentation and send it to the change team. The procedure for this is given in “Sending the documentation to the change team” on page 266.
3. Lastly, you need to apply the PTF resulting from the APAR, possibly after testing the fix on your system. This is described in “Applying the fix” on page 267.

Collecting the documentation for the APAR

As a general rule, the documentation you need to submit for an APAR includes all the material you need yourself to perform problem determination. Some of the documentation is common to all CICS problems, and some is specific to particular types of problem.

Make sure the problem you have described can be seen in the documentation you send. If the problem has ambiguous symptoms, you need to reveal the sequence of events leading up to the failure. Tracing is valuable in this respect, but you

© Copyright IBM Corp. 1979, 1999

might be able to provide details that trace cannot give. You are encouraged to annotate your documentation, if your annotation is legible and if it does not cover up vital information. You can highlight data in any hard copy you send, using transparent highlighting markers. You can also write notes in the margins, preferably using a red pen so that the notes are not overlooked.

Finally, note that if you send too little documentation, or if it is unreadable, the change team will return the APAR marked “insufficient documentation”. It is, therefore, worthwhile preparing your documentation carefully and sending everything relevant to the problem.

The general documentation is described in “General documentation needed for all problems with CICS.” However, these are only guidelines—you must find out from the support center precisely what documentation you need to send for your specific problem.

General documentation needed for all problems with CICS

The following is a list of the general documentation you might be asked to submit for an APAR:

- Any hard or soft copy illustrating the symptoms of the problem.
- A system dump of the CICS address space. Format the whole system dump if you plan to submit hard copy. Otherwise, you can just send the system dump data set on tape.
- A CICS trace. Auxiliary trace is best, but internal trace will do if you do not have an auxiliary trace. If internal trace has been running, it is in the system dump in any case.
- Listings of CICS management modules, such as DFHKCP, that are not supplied as object code only. **These should be unnecessary if preassembled modules are used**.
- Relevant CICS tables. Again, these should be unnecessary if preassembled versions are used.
- Listings of relevant application programs.
- Console logs.
- CICS logs (for example, the CSMT log) wherever possible. These contain information that is often overlooked. They are particularly useful when VTAM is in use.
- JCL listings. These may appear on dumps, and need not be sent twice.
- A list of PTFs and APARs applied.
- Details of any user modifications.

Sending the documentation to the change team

Follow the directions in APAR II02709 when gathering documentation to send to the change team. If you do not have access to the RETAIN database, ask your IBM representative to obtain a copy of APAR II02709 for you. The documentation you submit for the problem is best shipped in an APAR box, which you can obtain from your local IBM branch. APAR boxes are clearly marked as such, and they have a panel where tracking information such as the APAR number can be written.

Packing and mailing the APAR box

Place all your documentation and notes in one or more APAR boxes, making sure that the boxes are marked, for example, “1 of 2”, and so on, if you need to use more than one.

If you include any magnetic tapes, write this clearly on the outside of the box. This lessens the chance of their being stored in magnetic fields strong enough to damage the data.

To make sure the documentation reaches the correct destination, that is, the CICS change team, write the following on the box:

```
SHIP TO CODE 5U6
```

You also need a mailing label with the address of the CICS change team on it.

When the change team receives the package, this is noted in your APAR record on the RETAIN system. The team then investigates the problem. Occasionally, they need to ask the Support Center to contact you for more documentation, perhaps specifying some trap you must apply before getting it.

When the problem is solved, a code is entered on RETAIN to close the APAR, and you are provided with a fix.

You can enquire any time at your Support Center on how your APAR is progressing, particularly if it is a problem of high severity.

Applying the fix

When the change team have found a fix for your problem, they might want you to test it on your system. If they do ask you to test the fix, you are normally given two weeks to do it and to provide them with the results. However, you can ask for an extension if you are unable to complete the testing in that time.

When the team is confident that the fix is satisfactory, the APAR is certified by the CICS development team and the APAR is closed. You receive notification when this happens.

The APAR becomes a PTF

If the solution involves a change to code in a CICS module that you can assemble, you are sent the code change right away. The change is later distributed as a PTF.

If you cannot assemble the module yourself, because it involves a part of CICS that is object serviced, you might be supplied with a ZAP or a TOTEST PTF.

If you want a PTF to resolve a specific problem, you can order it explicitly by its PTF number through the IBM Support Center. Otherwise, you can wait for the PTF to be sent out on the standard distribution tape.

Part 5. Appendixes

© Copyright IBM Corp. 1979, 1999

Appendix A. SDUMP contents and INFOANA EXIT keywords

The following two tables provide a cross-reference between the CICS control blocks contained in an SDUMP and their associated system dump formatting keyword.

The first table provides a list of system dump formatting keywords and the CICS control blocks that they display.

The second table provides a list of all CICS control blocks in an SDUMP, alphabetically, with their associated system dump formatting keyword.

See “Formatting system dumps” on page 244 for a list of all possible system dump formatting keywords.

© Copyright IBM Corp. 1979, 1999

Finding the control blocks from the keywords

AI keyword

AP keyword

- CICS24 (task storage—below 16MB, CICS key)
- CICS31 (task storage—above 16MB, CICS key)
- DBL (journal control dynamic log buffer)
- DWE (user DWE storage)
- EIB (EXEC interface block)
- EIS (EXEC interface structure)
- EIUS (EXEC interface user structure)
- FILE (user file storage)
- JCA (journal control area)
- MAPCOPY (user BMS MAP storage)
- QCA (queue control area)
- QEA (queue element area)
- SYSEIB (system EXEC interface block)
- SYS_TCA (task control area—system area only)
- TD (user transient data)
- TS (user temporary storage)
- TCA (task control area—user)
- USER24 (task storage—below 16MB, user key)
- USER31 (task storage—above 16MB, user key)

CC keyword

- Local (anchor block)
  - LCBUFFER (local catalog buffers—one each thread)
  - LC_ACB (local catalog ACB)
  - LC_RPL (local catalog RPLs—one each thread)
- Global (anchor block)
  - GCBUFFER (global catalog buffers—one each thread)
  - GC_ACB (global catalog ACB)
  - GC_RPL (global catalog RPLs—one each thread)

CP keyword

- CP STATIC—Common Programming Interface (CPI) static storage

CSA keyword

- CSA (common system area)
- CSAOPFL (CSA optional features list)
- CWA (common work area)

DD keyword

- DDANCHOR (directory manager anchor block)
- For each directory:
  - DIR_HEAD (directory header)
  - AVL_HEDR (AVL header)
  - HASH_TBL (hash table)

DM keyword

- DMANCHOR (domain manager anchor block)
- WQP (domain wait queue)

DS keyword

- DSANC (dispatcher anchor block)
- DS_TCB (TCB block)
- DTA (dispatcher task area)
- SUSPAREA (unformatted SUSPEND_AREAS/TOKENS)
- TASK (unformatted DTAs)

DU keyword

- DUA (dump domain anchor block)
- DUBUFFER (transaction dump data set buffer)
- OPENBLOK (transaction dump Open block)
- SDTE (system dump table elements)
- TDTE (transaction dump table elements)

FCP keyword

- ACB (VSAM ACBs)
- AFCTE (application file control table element)
- DTF (DAM file DTF)
- DSNB (data set name block)
- DFHDTTABLE (data table base area)
- DFHDTFILE (data table path area)
- DFHDTHEADER (data table global area)
- DTRGLOBL (data table remote global area)
- FBWA (data table browse area)
- FCSTATIC (FCP static storage—anchor block)
- FCTE (file control table element)
- SHRCTL (shared LSRPOOLs)
- VSWA (VSAM work area)

ICP keyword

- ICE (interval control elements/AIDs)

JCP keyword

- DFHJCOCL (journal open/close parameter list)
- DFHJCT (journal control table header)
- DFHJCTTE (journal table entry)
- JAB (journal archive anchor block)
- JACD (journal archive control data set ACB)
- JACD (journal archive control data set RPL)
- JAW (journal archive work element)
- JCBUFFER (journal buffer)
- JCDTF (journal DTF)
- LECBS (logical ECBs)

KE keyword

- AFCB (CICS AFCB)
- AFCS (CICS AFCS)
- AFT (CICS AFT)
- AUTOSTCK (automatic storage stack entry)
- KCB (kernel anchor block)
- KERNSTCK (kernel linkage storage stack entry)
- KERRD (kernel error data)
- KTCB (KTCB table entry)
- DOH (domain table header)

- DOM (domain table entry)
- TAH (task table header)
- TAS (task table entry—TASENTRY)
- TCH (KTCB table header)

LD keyword

- APE (active program element)
- CPE (current program element)
- CSECTL (program CSECT List)
- LD_GLBL (loader domain global storage—anchor block)
- LDBE (loader domain browse element)
- LDWE (loader domain wait element)
- LLA (load list area)

LM keyword

- FREECHAI (LM domain freechain 1)
- FREECHAI (LM domain freechain 2)
- FREECHAI (LM domain freechain 3)
- LMANCHOR (lock manager domain anchor block)
- LMQUICK1 (LM domain quickcell 1)
- LMQUICK2 (LM domain quickcell 2)
- LMQUICK3 (LM domain quickcell 3)
- LOCK_ELE (LM domain lock element)

ME keyword

- MEA (message domain anchor block)

MN keyword

- MCT (monitoring control table)
- MNA (monitoring domain global storage—anchor block)
- MNAFB (monitor authorization facility parameter block)
- MNCONNS (monitor field connectors)
- MNDICT (monitor dictionary)
- MNDMF (DMF record buffer)
- MNEXC (exception record buffer)
- MNEXLIST (user EMP address list)
- MNFLDMAP (excluded/included CICS field map)
- MNPER (performance data buffer)
- MNTMA (transaction monitoring area)

MRO keyword

- CCB (connection control block)
- CRB (CICS region block)
- CSB (connection status block)
- DWE (deferred work element)
- LACB (logon address control block)
- LCB (logon control block)
- LXA (LX array)
- SCACB (subsystem connection address control block)
- SCCB (subsystem connection control block)
- SCTE (subsystem control table extension)
- SLCB (subsystem logon control block)
- SUDB (subsystem user definition block)
- UCA (use count array)

- URD (unit of recovery descriptor)

PA keyword

- DFHSIT (system initialization table)
- OVERSTOR (override parameter temporary work area)
- PAA (parameter manager domain anchor block)
- PARMSAVE (SIT override parameters)
- PRVMODS (SIT PRVMOD list)

PCP keyword

- PPTTE (program processing table entries)

PCT keyword

- PCTTE (program control table entries)

PG keyword

- PGA (program manager anchor)
- LLE (load list element)–can be system LLE or task LLE)
- PGWE (program manager wait element)
- PPTE (program processing table element)
- PTA (program transaction area)
- PLCB (program manager program level control block)
- HTB (handle table)

For an explanation of PG summary data in a level-1 dump, see Appendix B, “Summary data for PG and US keywords” on page 285.

RC keyword

- PSG (system spooling global control block)
- PSC (Continuous report control block)
- PST (CEMS operator task control block)
- PSW (Writer task control block)

RM keyword

- DFHRMCBS (recovery manager table storage)

SM keyword

- CTN (cartesian tree node)
- DXE (DSA extent list element)
- DXG (DSA extent GETMAIN description)
- DXH (DSA extent list header)
- MCA (SM macro-compatibility control area)
- PAM (page allocation map)
- PPA (page pool control area)
- PPX (page pool extent control area)
- QPF (quickcell page free element)
- QPH (quickcell page header)
- SAE (storage access table entry)
- SAT (storage access table)
- SCA (subpool control area)
- SCE (storage element descriptor)
- SCF (free storage descriptor)
- SMA (storage manager domain anchor block)
- SMSVCTRT (DFHSMSVC trace table)
- SMX (transaction storage area)

- SQE (suspend queue element)
- STAB (storage manager statistics buffer)

SSA keyword

- SSAL (static storage address list)
- SSA (static storage areas)

ST keyword

- STANCHOR (statistics domain anchor block)
- STSAFPB (statistics authorization facility parameter block)
- STDMF (statistics DMF record)
- STSTATS (statistics domain statistics record)

SZ keyword

- SZSDS (FEPI static area)

TCP keyword

- ACB (VTAM access method control block)
- AID (automatic initiation descriptor)
- AWE (autoinstall work element)
- BIND (bind image)
- DCB (SAM data control block)
- DIB (data interchange block)
- EXLST (VTAM ACB exit list)
- ISORM (indirect system object resolution map)
- LOGDS (extracted logon or CLSDST Pass data)
- NIB (node initialization block)
- PWE (postponed work element)
- RACE (receive-any control elements)
- RAIA (VTAM receive-any Input area)
- RPL (VTAM request parameter list—receive-any RPLs)
- SNTTE (SNT terminal entry)
- TACLE (terminal abnormal condition line entry)
- TCTENIB (NIB descriptor)
- TCTESBA (APPC send/receive buffer)
- TCTFX (TCT prefix)
- TCTTELUC (TCTTE APPC extension)
- TCTME (TCT mode entry)
- TCTSE (TCT system entries)
- TCTSK (TCT skeleton entries)
- TCTTE (TCT terminal entries)
- TCTTETTE (TCTTE extension)
- TCTTEUA (TCTTE user area)
- TIOA (terminal I/O area)
- WAITLST (wait list)
- ZEPD (TC module entry list)

TDP keyword

- ACB (TD VSAM ACB)
- BUFFER (TD I/O buffer)
- DCTE (destination control table entries)
- MBCA (TD buffer control area)
- MBCB (TD buffer control block)
- MQCB (TD queue control block)

- MRCA (TD string control area)
- MRCB (TD string control block)
- MRSD (TD CI state map—segment descriptor)
- MWCB (TD wait control block)
- RPL (TD VSAM RPL)
- SDSCI (TD SCSCI)
- TDST (TD static storage)
- VEMA (TD VSAM error message area)

TI keyword

- TIA (Timer domain anchor block)
- TRE (Timer request elements)

TMP keyword

- TMSTATIC (table manager static storage)
- SKT (scatter tables)
- DIRSEG (directory segments)
- TM_LOCKS (read lock blocks)

TR keyword

- TRA (trace domain anchor block)
- TRDCB (auxiliary trace data set DCB)
- TRDECB (auxiliary trace data set DECB)

TSP keyword

- TSACA (TS auxiliary control area)
- TSBMAP (TS byte map)
- TSBCA (TS buffer control area)
- TSBUFFER (TS I/O buffer)
- TSCOM (TS common area)
- TSGID (TS group identifier)
- TSRE (TS request element)
- TST (TS table header)
- TSTTE (TS table entry)
- TSUT (TS unit table header)
- TSUTE (TS unit table entry)
- TSVCA (TS VSWA control area)

UEH keyword

- DWE (deferred work element)
- EPB (exit program blocks)
- GWA (EPB global work area)
- TIE (task interface element)
- UET (user exit table)
- URD (unit of recovery descriptor)

US keyword

- USA (user domain anchor block)
- USXD (user domain transaction data)
- USUD (user domain user data), one or more of the following:
  - Principal
  - Session
  - EDF

For an explanation of US summary data in a level-1 dump, see Appendix B, “Summary data for PG and US keywords” on page 285.

XM keyword

- XMA (XM domain anchor block)
- TXN (XM domain transaction))
- TCL (XM domain tclass)
- XM_XB (XM domain browse element)
- MXT (XM domain MXT tclass)

XRF keyword

- XRPSTAT (XRP static storage)
- XRP_ACTS (XRP active status area)
- XRP_ALTS (XRP alternate status area)
- XRP_HLTH (XRP health area)
- XRP_XRSA (XRP anchor area)
- CAVM_STA (CAVM static storage)

XS keyword

- XSA (security domain anchor block)
- XSSS (security supervisor storage)

Finding the keywords from the control blocks

CICS Control Block		INFOANA DATA
ACB	VSAM ACBs	FCP
ACB	VTAM access method control block	TCP
ACB	TD VSAM ACB	TDP
AFCB	CICS AFCB	KE
AFCS	CICS AFCS	KE
AFCTE	application file control table element	FCP
AFT	CICS AFT	KE
AID	automatic initiation descriptor	TCP
AITM	static storage	AI
AITMTE	autoinstall terminal models	AI
APE	active program element	LD
AUTOSTCK	automatic storage stack entry	KE
AVL_HEDR	AVL tree header	DD
AWE	autoinstall work element	TCP
BIND	bind image	TCP
BUFFER	TD I/O buffer	TDP
CAVM_STA	CAVM static storage	XRF
CCB	connection control block	MRO
CCBUFFER	local catalog buffers, one each thread	CC
CICS24	task storage - below 16MB, CICS key	AP
CICS31	task storage - above 16MB, CICS key	AP
CPE	current program element	LD
CPSTATIC	common program interface storage	CP
CRB	CICS region block	MRO
CSA	common system area	CSA
CTN	cartesian tree node	SM
CSAOPFL	CSA optional features list	CSA

CICS Control Block		INFOANA DATA
CSB	connection status block	MRO
CSECTL	program CSECT list	LD
CWA	common work area	CSA
DBL	journal control dynamic log buffer	AP
DCB	data control block	FCP
DCB	data control block	JCP
DCB	SAM data control block	TCP
DCTE	destination control table entries	TDP
DDANCHOR	directory manager anchor block	DD
DECB	data extent control block	JCP
DFHJCOCL	journal open/close parameter list	JCP
DFHJCT	journal control table header	JCP
DFHJCTTE	journal table entry	JCP
DFHRMCBS	recovery manager table storage	RM
DFHSIT	system initialization table	PA
DIB	data interchange block	TCP
DIR_HEAD	directory header	DD
DIRSEG	directory segments	TMP
DMANCHOR	domain manager anchor block	DM
DOH	domain table header	KE
DOM	domain table entry	KE
DS_TCB	TCB block	DS
DSANC	dispatcher anchor block	DS
DSNB	data set name block	FCP
DFHDTTABLE	data table base area	FCP
DFHDTFILE	data table path area	FCP
DTA	dispatcher task area	DS
DFHDTHEADER	data table global area	FCP
DTRGLOBL	data table remote global area	FCP
DUA	dump domain anchor block	DU
DUBUFFER	transaction dump-data set buffer	DU
DWE	deferred work element	MRO
DWE	deferred work element	UEH
DWE	user DWE storage	AP
DXE	DSA extent list element	SM
DXG	DSA extent GETMAIN descriptor	SM
DXH	DSA extent list header	SM
EIB	EXEC interface block	AP
EIS	EXEC interface structure	AP
EIUS	EXEC interface user structure	AP
EPB	exit program blocks	UEH
EXLST	VTAM ACB exit list	TCP
FBWA	data table browse area	FCP
FCSTATIC	FCP static storage - anchor block	FCP
FCTE	file control table element	FCP
FILE	user file storage	AP
FREECHAI	LM domain freechain 1	LM
FREECHAI	LM domain freechain 2	LM
FREECHAI	LM domain freechain 3	LM

CICS Control Block		INFOANA DATA
GC_ACB	global catalog ACB	CC
GC_RPL	global catalog RPLs, one each thread	CC
GCBUFFER	global catalog buffers, one each thread	CC
GWA	EPB global work area	UEH
HASH_TBL	hash table	DD
HTB	handle table	PG
ICE	interval control elements/AIDs	ICP
ISORM	indirect system object resolution map	TCP
JAB	journal archive anchor block	JCP
JACD ACB	journal archive control data set ACB	JCP
JACD RPL	journal archive control data set RPL	JCP
JAW	journal archive work element	JCP
JCA	journal control area	AP
JCBUFFER	journal buffer	JCP
KCB	kernel anchor block	KE
KERNSTCK	kernel linkage storage stack entry	KE
KERRD	kernel error data	KE
KTCB	KTCB table entry)	KE
LACB	logon address control block)	MRO
LC_ACB	local catalog ACB	CC
LC_RPL	local catalog RPLs, one each thread	CC
LCB	logon control block)	MRO
LD_GLBL	loader domain global storage - anchor block	LD
LDBE	loader domain browse element	LD
LDWE	loader domain wait element	LD
LECBS	logical ECBs	JCP
LLE	load list element	PG
LMANCHOR	lock manager domain anchor block	LM
LMQUICK1	LM domain quickcell 1	LM
LMQUICK2	LM domain quickcell 2	LM
LMQUICK3	LM domain quickcell 3	LM
LOCK_ELE	LM domain lock element	LM
LOGDS	extracted logon or CLSDST pass data	TCP
LXA	LX array	MRO
MAPCOPY	user BMS MAP storage	AP
MBCA	TD buffer control area	TDP
MBCB	TD buffer control block	TDP
MCA	SM macro-compatibility control area	SM
MCT	monitoring control table	MN
MEA	message domain anchor block	ME
MNA	monitor domain global storage - anchor block	MN
MNAFB	monitor authorization facility parameter block	MN
MNCONNS	monitor field connectors	MN
MNDICT	monitor dictionary	MN
MNDMF	DMF record buffer	MN
MNEXC	exception record buffer	MN
MNEXLIST	user EMP address list	MN
MNFLDMAP	excluded/included CICS field map	MN
MNPER	performance data buffer	MN
MNTMA	transaction monitoring area	MN

CICS Control Block		INFOANA DATA
MQCB	TD queue control block	TDP
MRCA	TD string control area	TDP
MRCB	TD string control block	TDP
MRSD	TD CI state map - segment descriptor	TDP
MWCB	TD wait control block	TDP
MXT	XM domain MXT tclass	XM
NIB	node initialization block	TCP
OPENBLOK	transaction dump open block	DU
OVERSTOR	override parameter temporary work area	PA
PAA	parameter manager domain anchor block	PA
PAM	page allocation map	SM
PARMSAVE	SIT override parameters	PA
PCTTE	program control table entries	PCT
PGA	program management anchor	PG
PGWE	program management wait element	PG
PLCB	program management program control blk	PG
PPA	pagepool control area	SM
PPTE	program processing table element	PG
PPTTE	program processing table entries	PCP
PPX	pagepool extent control area	SM
PRSTATIC	partner resource static area	PR
PRTE	partner resource table entries	PR
PRVMODS	SIT PRVMOD list	PA
PSC	Continuous report control block	RC
PSG	system spooling global control block	RC
PST	CEMS operator task control block	RC
PSW	Writer task control block	RC
PTA	program transaction area	PG
PWE	postponed work element	TCP
QCA	queue control area	AP
QEA	queue element area	AP
QPH	quickcell page header	SM
QPK	quickcell page free element	SM
RACE	receive-any control elements	TCP
RAIA	VTAM receive-any input area	TCP
RMCBS	recovery manager control blocks	RM
RPL	VTAM request parameter list - receive-any RPLs	TCP
RPL	TD VSAM RPL	TDP
SAE	storage access table entry	SM
SAT	storage access table	SM
SCA	subpool control areas	SM
SCACB	subsystem connection address control block	MRO
SCCB	subsystem connection control block	MRO
SCE	storage element descriptor	SM
SCF	free storage descriptor	SM
SCTE	subsystem control table extension	MRO
SDSCI	TD SCSCI	TDP
SDTE	system dump table elements	DU
SHRCTL	LSRPOOLs	FCP
SKT	scatter tables	TMP
SLCB	subsystem logon control block	MRO

CICS Control Block		INFOANA DATA
SMA	storage manager domain anchor block	SM
SMX	transaction storage area	SM
SNTTE	SNT terminal entry	TCP
SQE	suspend queue element	SM
SSA	static storage areas	SSA
SSAL	static storage address list	SSA
STANCHOR	statistics domain anchor block	ST
STDMF	statistics DMF record	ST
STSAFPB	statistics authorization facility parameter block	ST
STSTATS	statistics domain statistics record	ST
SUDB	subsystem user definition block	MRO
SUSPAREA	unformatted SUSPEND_AREAS/TOKENS	DS
SYS_TCA	TCA - system	AP
SZSDS	FEPI static area	SZ
TACLE	terminal abnormal condition line entry	TCP
TAH	task table header	KE
TAS	task table entry - TASENTRY	KE
TASK	unformatted DTAs	DS
TCA	task control area	AP
TCH	KTCB table header	KE
TCL	XM domain tclass	XM
TCTENIB	NIB descriptor	TCP
TCTESBA	LU6.2 send/receive buffer	TCP
TCTFX	TCT prefix	TCP
TCTME	TCT mode entry	TCP
TCTSE	TCT system entries	TCP
TCTSK	TCT skeleton entries	TCP
TCTTE	TCT terminal entries	TCP
TCTTELUC	TCTTE LU6.2 extension	TCP
TCTTETTE	TCTTE extension	TCP
TCTTEUA	TCTTE user area	TCP
TD	user transient data	AP
TDST	TD static storage	TDP
TDTE	transaction dump table elements	DU
TIA	timer domain anchor block	TI
TIE	task interface element	UEH
TIOA	terminal I/O area	TCP
TM_LOCKS	read lock blocks	TMP
TMSTATIC	table manager static storage	TMP
TRA	trace domain anchor block	TR
TRDCB	auxiliary trace data set DCB	TR
TRDECB	auxiliary trace data set DECB	TR
TRE	timer request elements	TI
TS	user temporary storage	AP
TSACA	TS auxiliary control area	TSP
TSBCA	TS buffer control area	TSP
TSBMAP	TS byte map	TSP
TSBUFFER	TS I/O buffer	TSP
TSCOM	TS common area	TSP
TSGID	TS group identifier	TSP
TSRE	TS request element	TSP
TST	TS table header	TSP
TSTTE	TS table entry	TSP
TSUT	TS unit table header	TSP
TSUTE	TS unit table entry	TSP

CICS Control Block		INFOANA DATA
TSVCA	TS VSWA control area	TSP
TXN	XM domain transaction	XM
UCA	use count array	MRO
UET	user exit table	UEH
URD	unit of recovery descriptor	MRO
URD	unit of recovery descriptor	UEH
USER24	task storage - below 16MB, user key	AP
USER31	task storage - above 16MB, user key	AP
VEMA	TD VSAM error message area	TDP
VSWA	VSAM work area	FCP
WAITLST	wait list	TCP
WQP	domain wait queue	DM
XMA	XM domain anchor block	XM
XM_XB	XM domain browse element	XM
XRP_ACTS	XRP active status area	XRF
XRP_ALTS	XRP alternate status area	XRF
XRP_HLTH	XRP health area	XRF
XRP_XRSA	XRP anchor area	XRF
XRPSTAT	XRP static storage	XRF
ZEPD	TC module entry list	TCP

Appendix B. Summary data for PG and US keywords

This appendix lists the elements of the control block summaries in a level-1 analyzed dump for the PG and US keywords.

PG keyword

The summaries appear below in the sequence in which they appear in a dump. This is broadly the sequence in which the control blocks are listed in Appendix A, but note:

- The system LLE summary, if present, follows the PGA summary, but the task LLE summary, if present, follows the PTA summary.
- The HTB does not appear in a summary.

PGA (program manager anchor)

PG Domain Status
One of the following:

- Initializing
- Initialized
- Quiescing
- Quiesced
- Terminating
- Terminated

Autoinstall status
Either active or inactive

Autoinstall catlg status
Autoinstall catalog status, one of the following:

- All
- Modify
- None

Autoinstall exit name
Autoinstall exit name

Attempted autoinstalls
Number of attempted autoinstalls in decimal

Failed autoinstalls
Number of failed autoinstalls in decimal

Rejected autoinstalls
Number of rejected autoinstalls in decimal

XRSINDI active
Status of user exit, either Y or N

Exec calls allowed
Either Y or N

System LLE chain head
Address of system LLE chain head, zero if no chain exists

© Copyright IBM Corp. 1979, 1999

PGWE chain head
Address of PGWE chain head, or zero if no chain exists

Stats last - 1st word
Statistics last reset time using GMT (on two lines)

Reset time - 2nd word
Second part of last reset time

SM access token
SM access token value

SM isolation token
SM isolation token value

Storage protect
Either Y or N

Cold start
Either Y or N

Recovery complete
Either Y or N

System LLE Summary

LLE-ADDR
LLE address

PROGRAM
Program name

PPTE-ADD
PPTE address

PGWE Summary

PGWE-ADD
Address of suspended program

PROGRAM
Name of suspended program

SUS-TOKN
Suspend token

PPTE-ADD
Program PPTE address

PPTE Summary

PPTE ADDRESS
Address of PPTE block

PROGRAM NAME
The tables are indexed using the program name

MOD TYPE
Module type, one of the following:

- PG - Program
- MP - Mapset
- PT - Partitionset

LANG DEF

Language defined, one of the following:

- NDF - Not defined
- ASS - Assembler
- C - C
- COB - COBOL
- PLI - PL/I

LANG DED

Language deduced, one of the following:

- NDD - Not deduced
- ASS - Assembler
- C - C
- COB - COBOL
- COL - COBOL/VSE
- LEV - LEVSE
- PLI - PL/I

INST TYPE

PPTE installation type, one of the following:

- R - Built from RDO
- C - Built from catalog constant
- G - Built from grouplist
- A - Autoinstall
- S - System autoinstall
- M - Manual

CEDF STAT

CEDF status, either CED(CEDF allowed) or NOC(CEDF not allowed)

AVAL STAT

Program availability status, either E(enabled) or DI(disabled)

DATA LOC

Data location, either A (any location) or B(below 16M)

EXEC KEY

Execution key, either C (CICS) or U (user)

DPL SUBS

DPL subset, either DP (DPLsubset) or F (full API)

RE LOAD

Indicator of whether this is a reload program, either Y or N

LOAD STAT

Load status, one of the following:

- L - Loaded
- NL - Not loadable
- ND - Not loaded

HOLD STAT

CICS hold status, either C (loaded for CICS lifetime) or T (task lifetime)

USE COUNT

Use count in decimal, blank if 0

LOCK OWNER
Transaction number of locking program

PGWE CHAIN
Indicator of presence of any PGWEs, either Y or N.

REMOTE PRGID
Remote program name.

REMOTE SYSID
Remote system name.

REMOTE TRNID
Remote transaction name.

PTA Summary

TRAN NUM
Transaction number.

PTA ADDRESS
Address of PTA.

LOG-LVL
Logical level count in decimal.

SYS-LVL
System level count in decimal.

TASK-LLE
Address of task LLE head, zero if no task LLE exists.

PLCB
Address of PLCB head, or zero if no PLCB exists.

Task LLE Summary

LLE-ADDR
LLE address.

PROGRAM
Program name.

PPTE-ADD
PPTE address.

Task PLCB Summary

PLCB-ADD
PLCB address.

PROGRAM
Program name.

LOG-LVL
Logical level of program.

LOAD
Program load point.

ENTRY
Program entry point.

LENGTH
Program length.

CA-CURR
Current commarea address.

CLEN
Current commarea length.

INVK-PRG
Name of invoking program.

STG
Commarea storage class. Can be one of five:

- Blank - No commarea for this level.
- C - CICS
- C24 - CICS 24 bit
- U - User
- U24 - User 24 bit

EXIT-NME
Exit name derived from user exit number, if applicable.

ENV
Environment type, one of the following:

- EXEC - Command level application
- GLUE - Global user exit
- PLT - PLT program
- SYS - CICS system program
- TRUE - Task-related user exit
- URM - User-replaceable program

PPTE-ADD
Program PPTE address.

US keyword

A level-1 dump summarizes only the user domain data (USUD). The fields displayed are the same for each type of USUD (principal, session, or EDF).

USXD summary

TRAN NUM
Transaction number.

PRINCIPAL TOKEN
Principal token, if any.

SESSION TOKEN
Session token, if any.

EDF TOKEN
EDF token, if any.

USUD summary

TOKEN
User token.

USERID
User identifier.

GROUPID
Group identifier.

ADDCOUNT
Adduser use count.

TRNCOUNT
Transaction use count.

OPID
Operator identifier.

CLASSES
A bitmap expressing the operator classes in order 24 to 1.

PRTY
Operator priority.

TIMEOUT
Timeout interval in hours and minutes (hh:mm).

ACEE
Address of ACEE.

XRFSOFF
XRF user signon. Can be NOFORCE or FORCE.

USERNAME
User name.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

© Copyright IBM Corp. 1979, 1999

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

Bookmanager, CICS, CICS/VSE, IBM, NetView, RETAIN, VSE/ESA, VTAM

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
Release Guide	GC33-1645
Migration Guide	GC33-1646
Report Controller Planning Guide	GC33-1941
General	
Master Index	SC33-1648
Trace Entries	SC34-5556
User's Handbook	SC34-5555
Glossary (softcopy only)	GC33-1649
Administration	
System Definition Guide	SC33-1651
Customization Guide	SC33-1652
Resource Definition Guide	SC33-1653
Operations and Utilities Guide	SC33-1654
CICS-Supplied Transactions	SC33-1655
Programming	
Application Programming Guide	SC33-1657
Application Programming Reference	SC33-1658
Sample Applications Guide	SC33-1713
Application Migration Aid Guide	SC33-1943
System Programming Reference	SC33-1659
Distributed Transaction Programming Guide	SC33-1661
Front End Programming Interface User's Guide	SC33-1662
Diagnosis	
Problem Determination Guide	GC33-1663
Messages and Codes Vol 3 (softcopy only)	SC33-6799
Diagnosis Reference	LY33-6085
Data Areas	LY33-6086
Supplementary Data Areas	LY33-6087
Communication	
Intercommunication Guide	SC33-1665
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
Recovery and Restart Guide	SC33-1666
Performance Guide	SC33-1667
Shared Data Tables Guide	SC33-1668
Security Guide	SC33-1942
External CICS Interface	SC33-1669
XRF Guide	SC33-1671
Report Controller User's Guide	GC33-1940
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449
CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

© Copyright IBM Corp. 1979, 1999

Books from VSE/ESA 2.4 base program libraries

VSE/ESA Version 2 Release 4

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User's Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer's Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User's Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User's Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User's Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.4 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User's Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

Index

A

abbreviated-format trace 204
abend codes
  transaction 27
    AICA 29
    ASRA 29
    ASRB 29
    CICS 28
    destination 27
    documentation 28
    interpretation 28
    product other than CICS 28
    user 28
ABEND symptom keyword 9
abends
  AICA 130
  CICS system
    See CICS system abends
  codes
    See abend codes
  dump not made when expected 153
  exception trace entry 40
  investigating 39
    the documentation you need 39
  looking at the symptom string 41
  symptom keyword 9
  transaction 27
    See also transaction abends
    AICA 29
    ASRA 29
    ASRB 29
    worksheet 37
access methods
  determining the type in use 96
  intersystem communication 97
  possible reason for stalled system 127
ADD_SUSPEND function 60
  input parameters 60
addressing exception 32
AEYD
  causes 29
AICA abend
  probable cause 29
  PSW 238
  registers 238
AICA abends 130
AID (automatic initiation descriptor)
  See automatic initiate descriptor (AID)
AID chain
  investigating tasks that have not started 168
  locating in the formatted system dump 169
AIDTRMID (symbolic ID of terminal) 169
AIDTRNID (transaction ID) 169
AITM resource name 69
ALLOCATE resource type 69, 104
alternate system waits 121
ALTPAGE attribute 158, 162
ALTSCREEN attribute, 158, 162
Any_MBCB resource type 69
AP_INIT resource type 69
AP_QUIES resource type 69
AP_TERM resource type 69
APPC, range of devices 95
application programs
  dynamic storage 242
  storage areas 242
arithmetic exceptions 33
  investigating 33
ASIS option 158
ASRA abend
  causes 29
  execution key 238
  PSW 238
  registers 238
ASRB abend
  causes 29
  execution key 238
  PSW 238
  registers 238
ASRD abend
  causes 29
  PSW 238
  registers 238
assembler programs
  locating the DFHEISTG storage 242
assemblers
  errors in output 6
asynchronous processing 101
ASYNRESP resource name 69
ATI (automatic transaction initiation) 99
  See also automatic transaction initiation (ATI)
autoinitiated tasks
  excessive numbers shown in statistics 16
automatic initiate descriptor (AID)
  See also AID chain
  identifying the related task 169
  identifying the terminal 169
  investigating tasks that have not started 168
automatic transaction initiation (ATI)
  task produced no output 165, 168
    looking at the AID chain 168
    looking at the ICE chain 168
    resource not available 168
    task not yet scheduled to start 168

© Copyright IBM Corp. 1979, 1999

automatic transaction initiation session status 99
auxiliary trace
  abbreviated-format 204
  advantages 217
  characteristics 217
  controlling 218
  data sets 217
  destination 217
  DFHAUXT 217
  DFHBUXT 217
  extended-format 200
  formatting 199
    selectivity 199
  interpreting 200, 204
  loss of trace data 150
  status 217, 219
  switch status 217
  trace entries missing 152
AUXTR, system initialization parameter 218
AUXTRSW, system initialization parameter 218
AVAIL_nn resource name 71, 85

B

base locator linkage (COBOL)
  See BLL (base locator linkage (COBOL))
basic mapping support (BMS)
  See BMS (basic mapping support)
BATCH
  WAIT_EXTTERNAL input parameter 65
BLL (base locator linkage (COBOL))
  in COBOL dump 243
BMS (basic mapping support)
  applications not compiled with latest maps 162
  ASIS option 158
  attributes of fields 172
    DARK field attribute 172
  incorrect output to terminal 172
    attributes of fields 172
    DARK field attribute 172
    MDT 172
    modified data tag 172
    symbolic map 172
  maps incorrect 162
  MDT 172
  modified data tag 172
  symbolic map 172
bottlenecks 140, 146
  dispatch, suspend and resume cycle 141
  initial attach to the dispatcher 141
  initial attach to the transaction manager 140
  initial dispatch 141
BPS (builder parameter set)
  See builder parameter set (BPS)
builder parameter set (BPS)
  CSFE ZCQTRACE facility 235

C

CAVM (CICS availability manager) 221
  control data set 190
  message data set 190
CCSTWAIT resource type 69
CCVSAMWT resource type 69
CDSA resource type 69
CEBR transaction
  checking programming logic 171
  investigating loops 138
  use in investigating no task output 168
CECI transaction
  checking for bad data in a file 170
  checking programming logic 171
  investigating loops 138
  use in investigating no task output 168
CEDA in a wait state 122
CEDF transaction
  checking programming logic 171
  investigating loops 138
CEMT INQUIRE TASK
  HTYPE field 53
  HVALUE field 53
CEMT transaction
  SET PROGRAM NEWCOPY 6
  use during CICS termination 127
CEPWTERM resource type 69
CESN in a wait state 122
CETR transaction
  example screen 215, 218
  master system trace flag 213
  selecting components to be traced 214
  setting special trace levels 214
  setting standard trace levels 214
  suppressing standard tracing 213
  task tracing options 212
  terminal tracing options 212
  transaction tracing options 212
CHKSTRM option, startup override 178
CHKSTSK option, startup override 178
CICS availability manager
  See CAVM (CICS availability manager)
CICS availability manager (CAVM) 221
  trace table 221
CICS running slowly 11
CICS stalled
  caused by SOS condition 126
  during a run 125, 130
  during initialization 124
  during quiesce 127
  during termination 127
  effect of ICV parameter 125
  effect of ICVR parameter 125
  effect of ICVTSD parameter 126
  effect of MXT parameter 126

CICS stalled *(continued)*
investigating the reason for the stall 10
messages 10
on cold start 124
on emergency restart 124
on warm start 124
PLT initialization programs 124
PLT shutdown programs 128
possible causes 10
specific regions 16
system definition parameters wrong 125
CICS system abends
action of global trap/trace exit 254
CICS system dump following 227
CSMT log messages 11
exception trace entry 40
information needed by the Support Center 39
investigating 39
looking at the symptom string 41
messages 11
the documentation you need 39
CICS system dumps
data not formatted correctly 156
destination 236
dispatcher domain storage areas 83
dump not made on CICS system abend 153
following CICS system abend 227
following transaction abend 227
formatting 244
selectivity 244
from global trap/trace exit 254
global suppression 154, 225
in problem determination 225
internal trace table 40
investigating CICS system abends 39
investigating waits 52, 54
kernel domain storage areas
CICS system abends 40
error code 44
error data 46
error type 44
failing program 45
information provided 42
kernel error number 44
point of failure 45
PSW at time of error 46
registers at time of error 46
task error information 44
task summary 42
tasks in error 42
waits for resource locks 83
locating the AID chain 169
locating the ICE chain 169
lock manager domain storage areas 83
looking at the symptom string 41
of remote CICS regions 102
CICS system dumps *(continued)*
precautions using dump formatting keywords 156
statistics 230
storage manager domain storage areas 90
storage violation 176, 177, 182
suppression by user exit program 154
suppression for individual transactions 154, 225
suppression for specific dump codes 225
system dump code option 229
system dump codes 226
temporary storage control blocks 92
terminal control storage areas 96
CICS, resource name 73
CICS410 dump exit
classification of problems 9
COBOL programs
BLL (base locator linkage) 243
task global table 242
working storage 242
common system area (CSA)
in transaction dump 239
locating the AID chain 169
optional features list 239
compilers
errors in output 6
COMPLETION_CODE
RESUME input parameter 64
SUSPEND output parameter 63
component tracing
identifying codes 214
precautions when selecting 151
setting special trace levels 214
setting standard trace levels 214
CONSP resource type 69
control data set 190
control interval (CI)
exclusive control deadlock 112, 113
exclusive control waits 112
CPI resource name 69
CRTE and uppercase translation 160
CSA (common system area)
See common system area (CSA)
CSADLECB resource name 69
CSAOPFL 239
CSAOPFRC resource name 74
CSASSI2 resource name 69
CSATODTU 116
CSFE DEBUG transaction
global trap/trace exit 253
storage checking 178
TRAP operand 253
CSFE transaction
checking the programming logic 171
storage freeze option 171
CSFE ZCQTRACE transaction
dumps of builder parameter set 235

CSMT log
abend messages 3, 11, 13
terminal error messages 13, 95
CSNC resource type 71
CURRENTDDS, transaction dump data set status 236
CWA (common work area) 239

D

DAM
data corruption
bad programming logic 171
incorrect mapping to program 170
incorrect mapping to terminal 172
attributes of fields 172
DARK field attribute 172
MDT 172
modified data tag 172
symbolic map 172
incorrect records in file 170
missing records in file 170
possible causes 169
data exception 31
DATABUFFERS parameter of FILE resource definition 109
DBUGUSER resource name 70
DCT (destination control table)
See destination control table (DCT)
DCT resource name 74
deadlock time-out interval
description 67
EXEC CICS WRITEQ TS command 91
interval control waits 115
task storage waits 90
DEADLOCK_ACTION
description of parameter 67
SUSPEND input parameter 62
WAIT_EXTERNAL input parameter 65
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
debugging
IRC problems 195
multiregion operation problems 195
DELETE_SUSPEND function of gate DSSR 61
input parameters 61
output parameters 62
destination control table (DCT)
extrapartition transient data destinations 106
logically recoverable queues 106
DFGTRAP 253
DFH$AXRO 189
DFHAIIN resource type 69
DFHAUXT 217
DFHBUXT 217
DFHCPIN resource type 69
DFHDMPA dump data set 235
DFHDMPB dump data set 235
DFHDU410 job log 237
DFHEISTG 242
DFHEVBBF resource name 75
DFHJ1Snn resource name 73
DFHJAP resource name 71
DFHKC TYPE=DEQ macro 89
DFHKC TYPE=ENQ macro 88
DFHKC TYPE=WAIT macro 88
DCI=CICS option 88
DCI=LIST option 88
DCI=SINGLE option 88
DCI=TERMINAL option 88
DFHPRIN resource type 70
DFHPSPCK resource name 69
DFHPSPIO 73
DFHPSPIO resource name 69, 73, 74, 77
DFHRC5482 message 78
DFHSIPLT resource type 70
DFHSPSPIO resource name 75
DFHTACB 240, 242
PSW 240
registers 240
DFHTRADS DSECT 254
DFHXRA resource name 76
DFHXRCP resource name 75
DFHZARER resource name 76
DFHZARL1 resource name 76
DFHZARL2 resource name 77
DFHZARL3 resource name 77
DFHZARL4 resource name 77
DFHZARQ1 resource name 77
DFHZARR1 resource name 77
DFHZCRQ1 resource name 76
DFHZDSP resource name 74
DFHZEMU1 resource name 76
DFHZERH1 resource name 76
DFHZERH2 resource name 76
DFHZERH3 resource name 76
DFHZERH4 resource name 76
DFHZIS11 resource name 76
DFHZNAC1 resource name 76
DFHZRAQ1 resource name 76
DFHZRAR1 resource name 76
DISASTER value of dispatcher RESPONSE output parameter 67
dispatcher
ADD_SUSPEND function 60
DELETE_SUSPEND function 61
dispatch, suspend and resume cycle 141, 146
failure of tasks to get attached 141, 142
failure of tasks to get initial dispatch 141, 144
functions of gate DSSR 59
INQUIRE_SUSPEND_TOKEN function 61
RESUME function 64

dispatcher *(continued)*
SUSPEND function 62
suspension and resumption of tasks 59
tracing the suspension and resumption of tasks 54
WAIT_EXTERNAL function 64, 65
WAIT_OLDC function 64, 65
WAIT_OLDW function 64, 65
distributed transaction processing (DTP) 101
DL/I
DMB (data management block)
DMWTQUEU resource name 69
domain identifying codes 214
DS_NUDGE resource name 75
DS_SDJnn resource name 72
DS_SW_nn resource name 72
DSA (dynamic storage area)
current free space 91
storage fragmentation 91
DSSR gate of dispatcher domain
ADD_SUSPEND function 60
DELETE_SUSPEND function 61
INQUIRE_SUSPEND_TOKEN function 61
RESUME function 64
SUSPEND function 62
tracing the functions 54
interpreting the trace table 54
tracing the input and output parameters 54
WAIT_EXTERNAL function 64, 65
WAIT_OLDC function 64, 65
WAIT_OLDW function 64, 65
dump codes
checking the attributes 155
DUMPSCOPE option 227
RELATED attribute 227
storage violation 176
suppression of system dumps 225
suppression of transaction dumps 225
system 155, 226
CICS termination option 229
maximum dumps option 229
NOSYSDUMP attribute 155
options 229
SYSDUMP attribute 155
system dumping option 229
transaction 155, 226
CICS termination option 229
format 226
maximum dumps option 229
NOTRANDUMP attribute 155
options 229
system dumping option 229
TRANDUMP attribute 155
transaction dumping option 229
dump data sets
attributes 235
AUTOSWITCH status 236
dump data sets *(continued)*
CLOSED status 236
current status 236
DFHDMPA 235
DFHDMPB 235
inquiring on 235
NOAUTOSWITCH status 236
OPEN status 236
setting 235
switch status 236
dump domain
XDUREQ global user exit 154
dump table
examples 230, 232
options 155, 229
loss of additions and changes 229
preservation of additions and changes 229
statistics 230
current count 230
reset 230
system dumps suppressed 230
system dumps taken 230
times dump code action taken 230
transaction dumps suppressed 230
transaction dumps taken 230
suppression of dumping 155
system 232
temporary entries 229, 233
transaction 230
DUMP, system initialization parameter 154, 225
DUMPDS, system initialization parameter 236
dumps
CICS system
See CICS system dumps
controlling
CEMT transaction 226
dump codes 226
dump tables 226
examples 230, 232
EXEC CICS commands 226
selective dumping of storage 235
specifying dump options 229
using an undefined dump code 233
controlling dump action 225
current dump ID 232
dump output is incorrect
data not formatted correctly 156
dump not made on abend 153
investigating 153
some dump IDs missing from the sequence of dumps 155
wrong CICS partition 153
dumps 20
events that can cause dumps 226
IDs missing from the sequence of dumps 155
in problem determination 225

dumps *(continued)*
looking at the symptom string 41
options 229
requesting dumps 226
setting the dumping environment 225
suppressing 154
transaction
See transaction dumps
DUMPSCOPE dump code option 227
DUMPSW, system initialization parameter 236
dynamic storage area (DSA)
See DSA (dynamic storage area)

E

EARLYPLT resource name 70
ECB (event control block)
EXEC CICS POST command 116
function WAIT_EXTERNAL of gate DSSR 64, 65
function WAIT_OLDC of gate DSSR 64, 65
function WAIT_OLDW of gate DSSR 64, 65
invalid address, task control waits 89
posting after task is canceled 181
storage violations 181
valid address, task control waits 89
ECB_ADDRESS
WAIT_EXTERNAL input parameter 65
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
ECB_LIST_ADDRESS
WAIT_EXTERNAL input parameter 65
WAIT_OLDW input parameter 65
ECBTCP resource name 69
ECDSA resource type 70
EDF (execution diagnostic facility)
investigating loops 138
use in investigating no task output 166
EDF resource type 70
EDSA (extended dynamic storage area)
current free space 91
storage fragmentation 91
EIB (EXEC interface block)
See EXEC interface block (EIB)
EIBFN 239
in last command identification 241
EKCWAIT resource type 70
EMP (event monitoring point) 21
ENQUEUE on single server resource 89
Environmental Record Editing and Printing Program (EREP) 24
ERDSA resource type 70
EREP (Environmental Record Editing and Printing Program) 24
error code 44
error data 46
error number 44
error type 44
ESDSA resource type 70
EUDSA resource type 70
event control block (ECB)
See ECB (event control block)
event monitoring point (EMP) 21
exception trace
characteristics 220
CICS system abends 40
destination 220
format 220
missing trace entries 152
purpose 220
storage violation 176, 177, 180
user 220
EXCEPTION value of dispatcher RESPONSE output parameter 67
EXCLOGER resource name 69
EXEC CICS ABEND command 227
EXEC CICS DELAY command 115
EXEC CICS DUMP TRANSACTION command 226, 235
EXEC CICS ENTER TRACENUM command 220
EXEC CICS INQUIRE TASK
SUSPENDTYPE field 53
SUSPENDVALUE field 53
EXEC CICS PERFORM DUMP command 226
EXEC CICS POST 116
EXEC CICS READ UPDATE command 113
EXEC CICS RETRIEVE WAIT command 115
EXEC CICS REWRITE command 113
EXEC CICS START command 115, 141, 142
EXEC CICS STARTBR command 113
EXEC CICS WAIT EVENT command 116
EXEC CICS WRITE command 113
EXEC CICS WRITE MASSINSERT command 113
EXEC CICS WRITEQ TS command 91
NOSUSPEND 91
REWRITE option 91
EXEC interface block (EIB)
EIBFN 239
execution diagnostic facility (EDF)
investigating loops 138
execution exception 31
Execution key 238
exit programming interface (XPI)
ADD_SUSPEND function 60
correctness of input parameters 4
DELETE_SUSPEND function of dispatcher 61
need to observe protocols and restrictions 4
problems using 4
restrictions in user exits 4
RESUME function of dispatcher 64
SUSPEND function of dispatcher 62
suspension and resumption of tasks 59

exit programming interface (XPI) *(continued)*
SYSTEM_DUMP call 226
TRANSACTION_DUMP call 226
WAIT_EXTERNAL function of dispatcher 65
extended dynamic storage area
See EDSA (extended dynamic storage area)
extended-format trace 200
EXTENDEDDS attribute, TYPETERM 158, 162
extrapartition transient data waits 105

F

FCBFWAIT resource type 70, 109
FCCIWAIT resource type 71
FCDWWAIT resource type 70, 110
FCFRWAIT resource type 70
FCFSWAIT resource type 70, 110
FCINWAIT resource type 70
FCIOWAIT resource type 70, 110
FCPSWAIT resource type 70, 111
FCRBWAIT resource type 70, 111
FCSRSUSP resource type 70, 111
FCTISUSP resource type 70, 111
FCXCWAIT resource type 70, 112
FEPI
See front end programming interface
file accesses, excessive 16
file control waits 108
exclusive control deadlock 112, 113
file ID resource name 70
files
no task output 168
first failure data capture 216, 220
FLUSH_nn resource name 71, 85
FOREVER resource type 71
formatting CICS system dumps 244
front end programming interface (FEPI)
dump control option 249
FEPI waits 123
function shipping 101

G

global catalog data set (GCD)
dump table options 229
effect of redefinition on dump table 229
global trap/trace exit 182, 253
actions the exit can take 254
activating and deactivating the exit 253
coding 255
establishing the exit 253
information passed to the exit 254
program check handling 255
replacing a trap exit 253
uses 253
work area 255

H

HTYPE field 53
HVALUE field 53

I

I/O buffers, transient data
all in use 107
IBM Support Center
call receipt 259, 260
dealing with the Center 259
problem reporting 259
structure 262
use of RETAIN database 9, 261
when to contact 259
ICE (interval control element)
See interval control element (ICE)
ICE chain
ICE expiration 116
ICETRNID transaction ID 169
ICEXPIRY resource type 71
ICEXTOD expiration time 169
ICEXTOD value 116
ICGTWAIT resource type 71, 94, 115
ICMIDNTE resource type 71
ICV, system initialization parameter
possible cause of CICS stall 125
ICVR, system initialization parameter
non-yielding loops 130
possible cause of CICS stall 125
tight loops 130
ICVTSD, system initialization parameter
possible cause of CICS stall 126
ICWAIT resource type 71, 94, 115
incorrect output
abend codes 19
application did not work as expected 163
BMS mapping 172
attributes of fields 172
DARK field attribute 172
MDT 172
modified data tag 172
symbolic map 172
change log 20
checking for bad data in a file 170
checking the mapping from file to program 170
checking the programming logic 171
desk checking 171
using CEBR 171
using CECI 171
using CEDF 171
using interactive tools 171
databases 23
error messages 19
files 23

incorrect output *(continued)*
incorrect output read from VSAM data set 163
investigating 149
link-edit maps 19
manuals 19
monitoring 21
no output obtained 164
ATI tasks 165, 168
disabling the transaction 168
explanatory messages 164
finding if the task ran 166
looking at files 168
looking at temporary storage queues 168
looking at transient data queues 168
possible causes 164
START PRINTER bit on write control character 165
task not in system 165
task still in system 165
testing the terminal status 164
using CECI 168
using execution diagnostic facility 166
using statistics 167, 168
using trace 166
passed information 23
printed output wrong 156
source listings 19
statistics 21
symptom keyword 9
symptoms 13
temporary storage 22
terminal data 22
terminal output wrong 156
trace 23
trace data wrong 150
trace destination wrong 149
trace entries missing 151
trace output wrong 149
transaction inputs and outputs 22
transient data 22
unexpected messages 13
user documentation 19
wrong CICS components being traced 151
wrong output obtained 169
possible causes 169
wrong tasks being traced 151
INCORROUT symptom keyword 9
INDEXBUFFERS parameter of FILE resource definition 109
information sources 19
INFORMATION/ACCESS licensed program 9
initialization
See system initialization
initialization stall 124
INQUIRE_SUSPEND_TOKEN function of gate DSSR 61
INQUIRE_SUSPEND_TOKEN function of gate DSSR *(continued)*
output parameters 61
internal trace
abbreviated-format 204
characteristics 216
controlling 218
destination 216
exception trace destination 216
extended-format 200
formatting 199
interpreting 200, 204
status 216, 219
trace entries missing 152
trace table size 216
changing the size dynamically 216
wrapping 216
interregion communication
See IRC (interregion communication)
intersystem communication (ISC)
poor performance 142
waits 101, 104
INTERVAL
description of parameter 67
SUSPEND input parameter 62
WAIT_EXTERNAL input parameter 65
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
interval control
element 116
performance considerations 141
waits 115
deadlock time-out interval 115
systematic investigation 116
using trace 117
interval control element (ICE) 168
See also ICE chain
INTTR, system initialization parameter 218
INVALID value of dispatcher RESPONSE output parameter 67
IRC (interregion communication) 195
poor performance 142
waits 101, 104
IRLINK resource type 71, 94, 101
ISC (intersystem communication)
See intersystem communication (ISC)

J

JABSUTOK resource name 72
JACDIORB resource type 71
JASTMECB resource name 71, 85
JASUBTAS resource type 71, 85
JCAVLECB resource type 71, 85
JCBUFFER resource type 71, 85

JCCLDONE resource type 71
JCDETACH resource type 71
JCFLBUFF resource type 71, 85
JCINITN resource type 72
JCIOBLOK resource type 72, 85
JCIOCOMP resource type 72, 86
JCJASUS resource type 72
JCJOURDS resource type 72
JCLASTBK resource type 72, 86
JCOPDONE resource type 72, 86
JCREADY resource type 72
JCRQDONE resource type 72, 87
JCSWITCH resource type 72, 87
JCTAPE2 resource type 72, 87
JCTBAECB resource name 71
JCTERMN resource type 73
JCTICA resource name 72
JCTIMER resource type 73
JCTIOECB resource name 72, 86
JCTXAECB resource name 72, 86
JCTXBECB resource name 72, 86
Jnnbbbbb resource name 72, 85, 86
journal control
  contention for exclusive control of journal 85
  waits 85
JOURNAL resource name 72

K

Katakana terminals 161
  mixed English and Katakana characters 161
KC_ENQ resource type 73, 88, 89, 110
  DAM record locking 115
  VSAM record locking 114
KCCOMPAT resource type 73, 88
  resource names 88
    CICS 88
    LIST 88
    SINGLE 88
    SUSPEND 88
    TERMINAL 88, 94
kernel domain
  information given in dump 42
    error code 44
    error data 46
    error table 44
    error type 44
    failing program 45, 47
    kernel error number 44
    point of failure 45
    PSW at time of error 46
    registers at time of error 46
    storage addressed by PSW 46
    storage addressed by registers 46
    task error information 44
    task summary 42
    tasks in error 42, 44
kernel domain *(continued)*
  linkage stack 47
    identifying the task in error 47
  stack entries 239
  storage areas 42
KERNEL resource type 73
KERNERROR value of dispatcher RESPONSE output parameter 67

L

last command identification 241
last statement identification 242
LATE_PLT resource name 70
LECBECB resource name 71, 85
level-1 trace points 207
level-2 trace points 208
level-3 trace points 208
link editor
  errors in output 6
LIST resource name 73, 77
LMQUEUE resource name 69, 82
loader domain (LD)
  program storage map 49
  waits 108
lock manager domain (LM)
  involvement in waits 82
    identifying the lock owning task 83
    investigating 83
logon rejection 157
LOOP symptom keyword 9
loops
  CICS detected 129
  debugging with interactive tools 138
  identifying the point of entry 135
  in CICS code 129
  in system code 136
  investigating 129
    looking at the evidence 135
    techniques 134, 136
  investigating by modifying your program 138
  looking at the transaction dump 136
  non-yielding 129
    characteristics 130
    finding the cause 136
    investigating 134
    possible causes 136
  possible causes 129
  potential consumption of storage 91
  potential consumption of temporary storage 92
  symptom keyword 9
  symptoms 14, 15, 129
    CICS region stalled 16
    CPU usage high 16
    reduced activity at terminals 16
    repetitive output 16
    short on storage 16

loops *(continued)*
symptoms *(continued)*
system busy symbol 16
tight 129
characteristics 130
finding the cause 136
identifying an instruction in the loop 136
investigating 134
possible causes 136
types 129
using CEBR 138
using CECI 138
using CEDF 138
using the CEMT transaction 16
using trace 134, 136
yielding 129
characteristics 132
finding the cause 138
investigating 136
possible causes 138
useful documentation 137
lowercase characters 158
LUTYPE6.1, range of devices 95

M

MBCB_xxx resource type 73
message data set 190
MESSAGE symptom keyword 9
messages
absence of, when expected 16
CICS under stress 17
destination
CSMT 11, 13
CSTL 13
DFHAC2008 168
DFHSM0131 17, 145
DFHSM0133 17, 145
DFHSR0601 11
DFHST0001 11
dump formatting error 177
preliminary checks 3
short on storage 145
sources 5
storage violation 174, 177
symptom keyword 9
terminal errors 95
transaction abend 27
transaction disabled 168
unexpected 13, 158
missing trace entries 151
module index
in transaction dump 241
MONITOR POINT command 21
monitoring point 21
MRO waits 101, 104
MROQUEUE resource name 71
multiregion operation using IRC 195
multiregion operation waits 101, 104
MXT (maximum tasks value)
effect on performance 142
kernel task summary 43
possible cause of CICS stall 126
reason for task failing to start 12
waits 78
XM_HELD resource type 78
MXT resource type 73

N

NetView 128
networks
messages 95
preliminary checks 5
NOSYSDUMP, system dump code attribute 155
NOTRANDUMP, transaction dump code attribute 155

O

OK value of dispatcher RESPONSE output parameter 67
OLPD (Online Problem Determination) program
Online Problem Determination (OLPD) program
OPEN_ANY resource type 73
OPEN_NFY resource type 73
operation exceptions 31
output
absence when it is expected 14
incorrect
See incorrect output
none obtained 164
ATI tasks 165, 168
disabling the transaction 168
explanatory messages 164
finding if the task ran 166
looking at files 168
looking at temporary storage queues 168
looking at transient data queues 168
possible causes 164
START PRINTER bit on write control character 165
task not in system 165
task still in system 165
testing the terminal status 164
using CECI 168
using execution diagnostic facility 166
using statistics 167, 168
using trace 166
repetitive 16
wrong 169
possible causes 169

overseer sample program 189

P

PAGESIZE attribute, TYPETERM 162
PC, communication with CICS 95
PERFM symptom keyword 9
performance
  bottlenecks 140, 146
    dispatch, suspend and resume cycle 141
    initial attach to the dispatcher 141
    initial attach to the transaction manager 140
    initial dispatch 141
  dispatch, suspend and resume cycle 146
  initial attach to the dispatcher 142
  initial attach to the transaction manager 141
  initial dispatch to the dispatcher 144
  interval control delays 141
  MXT limit 142
  performance class monitoring 144
  poor
    at peak system load times 11
    finding the bottleneck 140
    investigating 139
    lightly loaded system 11
    possible causes 11
    symptom keyword 9
    symptoms 11, 14, 17, 139
  remote system status 142
  system loading 146
  task control statistics 142
  task priority 144
  task time-out interval 146
  terminal status 141
  using trace 143
performance class monitoring 144
PIC (program interrupt code)
  See program interrupt code (PIC)
PL/I application programs
  locating the DSA chain 242
PLT (program list table)
  See program list table (PLT)
PMR (problem management record) 261
poor performance
  See performance, poor
preliminary checks
  all functions fully exercised 6
  any changes to the application 6
  any previous success 3, 6
  changes since last success 4
    hardware modification 5
    initialization procedure 5
    modified application 5
    new application 5
    PTF (program temporary fix) 4
    resource definitions 5
preliminary checks *(continued)*
  common programming errors 7
  failure at specific times of day 4
  intermittent failures 4
  interval control waits 115
  messages 3
  network related errors 5
    many terminals 6
    single terminal 5
  no previous success 6
  output from assembler 6
  output from compiler 6
  output from link editor 6
  output from translator 6
  reproducible problems 3
    caused by poor system definition 4
    related to applications 3
    related to system loading 4
  terminal waits 95
PREOPNnn resource name 72, 87
printers 78
  no output 165
  printed output wrong 156, 161
  unexpected line feeds and form feeds 162
  waits 78
  write control character 165
privileged operation 31
PRM resource name 70
problem classification 9
problem determination
  FEPI waits 123
problem management record (PMR) 261
problem reporting
  documentation needed 261
  IBM Program Support 259
  information needed 260
  report sheet 259
processors
  usage high 16
PROFILE definition
  SCRNSIZE attribute 158, 162
  UCTRAN attribute 158
program check
  addressing exception 32
  arithmetic exceptions 33
    investigating 33
  cause of ASRA abends 29
  data exception 31
  execution exception 31
  investigating 30
  next sequential instruction 30
  operation exception 31
  outside of CICS 30
  possible types 31
  privileged operation 31
  protection exception 32

program check *(continued)*
specification exception 32
system interrupts 33
wild branch 31
program check and abend tracing 224
program control waits 108
program interrupt code (PIC)
addressing exception 32
arithmetic exceptions 33
data exception 31
execution exception 31
interpretation 31
operation exception 31
privileged operation 31
protection exception 32
specification exception 32
system interrupts 33
program list table (PLT)
programs executing during CICS quiesce 128
transient data waits 105, 124
program name resource name 74
PROGRAM resource type 73, 74
program status word (PSW) 29
See also PSW (program status word)
program support
See IBM Support Center
programming errors
preliminary checks 7
programs
information for current transaction 240
loops 129
problems with loading 108
representation in linkage stack 47
storage 240
protection exception 32
dealing with 33
possible causes 34
PS_INIT resource type 74
PSB (program specification block)
PSINQECB resource name 76
PSOP1ECB resource name 76
PSOP2ECB resource name 76
PSUNBECB resource name 76
PSW (program status word)
at time of error 46
CICS system abends 46
description 29
finding the offset of a failing instruction 49
format 49
in transaction abend control block 240
in transaction dump 238
PTF 267
PTF level 49
PURGEABLE operand
SUSPEND input parameter 62
WAIT_EXTERNAL input parameter 65
PURGEABLE operand *(continued)*
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
PURGED value of dispatcher RESPONSE output parameter 67

Q

quiesce stall 127

R

RDBLOKnn resource name 72
RDO transaction
See resource definition online (RDO)
RDRECDnn resource name 72
RDSA resource type 74
REASON
ADD_SUSPEND output parameter 60
DELETE_SUSPEND output parameter 62
RESUME output parameter 64
SUSPEND output parameter 63
WAIT_EXTERNAL output parameter 66
WAIT_OLDC output parameter 66
WAIT_OLDW output parameter 66
RECEIVE resource type 74
record locking
DAM data sets 115
VSAM data sets 114
registers
at time of error 46
CICS system abends 46
data addressed at the time of error 46
in transaction abend control block 240
in transaction dump 238
registers at last EXEC command 238
RELATED dump code attribute 227
Remote abend indicator 238
resource definition online (RDO)
ALTER mapset 6
ALTER program 6
ALTER transaction 6
DEFINE mapset 6
DEFINE program 6
DEFINE transaction 6
INSTALL option 6
resource names 77
AITM 69
ASYNRESP 69
AVAIL_nn 71, 85
CICS 73
CPI 69
CSADLECB 69
CSAOPFRC 74
CSASSI2 69
DBUGUSER 70

resource names *(continued)*
DCT 74, 105
DETACH 73
DFHEVBBF 75
DFHJ1Snn 73
DFHPSPCK 69
DFHPSPIO 69, 73, 74, 75, 77
DFHXMTA 71
DFHXRA 76
DFHXRCP 75
DFHZARER 76
DFHZARL1 76
DFHZARL2 77, 104
DFHZARL3 77, 104
DFHZARL4 77
DFHZARQ1 77
DFHZARR1 77
DFHZCRQ1 76, 103
DFHZDSP 74
DFHZEMU1 76
DFHZEMW1 103
DFHZERH1 76
DFHZERH2 76
DFHZERH3 76
DFHZERH4 76, 104
DFHZIS11 76
DFHZNAC1 76
DFHZRAQ1 76, 103
DFHZRAR1 76, 103
DLBL filename 72
DMWTQUEU 69
DS_NUDGE 75
DS_SW_nn 72
EARLYPLT 70
EXCLOGER 69
file ID 70
FLUSH_nn 71, 85
HVALUE 53
inquiring during task waits 53
JABSUTOK 72
JASTMECB 71, 85
JC_SDJnn 72
JCTBAECB 71
JCTICA 72, 86
JCTIOECB 72, 86
JCTXAECB 72, 86
JCTXBECB 72, 86
Jnnbbbbb 72, 85, 86
JOURNAL 72
LATE_PLT 70
LECBECB 71, 85
LIST 73, 77
LMQUEUE 69, 82
message queue ID 76
module name 71
MROQUEUE 71
resource names *(continued)*
PREOPNnn 72, 87
PRM 70
program ID 73
program name 74
program_name 73
PSINQECB 76
PSOP1ECB 76
PSOP2ECB 76
PSUNBECB 76
RDBLOKnn 72
RDRECDnn 72
SHUTECB 69
SINGLE 70, 73
SIPDMTEC 69
SISUBECB 74
SMSYSTEM 74
STABLEnn 72, 87
STARTUP 75
STATIC 70
STP_DONE 69
SUBTASK 71, 72, 73, 86, 87
summary of possible values 68
SUSPEND 73
SUSPENDVALUE 53
SYSIDNT/session ID 71
TCLASS 74
TCTTETI value 69
TCTVCECB 69, 76
TERMINAL 73
terminal ID 71
transaction ID 75
transient data queue name 69, 73, 74, 75, 105, 106, 107
TSBUFFER 75
TSEXTEND 75
TSIO 75
TSMCPECB 74
TSOPEN4B 75
TSQUEUE 75
TSSTRING 75
TSUT 75
TSWBUFFR 75
VSMSTRNG 69
WCSTECB 75
WTO 74
XM_HELD 73
XRSSTECB 76
ZC_ZGRP 74
ZSLSECB1 76
resource types 77, 78
ALLOCATE 69, 104
Any_MBCB 69, 107
Any_MRCB 69, 107
AP_INIT 69
AP_QUIES 69

resource types *(continued)*
AP_TERM 69
CCSTWAIT 69
CCVSAMWT 69
CDSA 69, 90
CEPWTERM 69
CONSP 69, 77
CSNC 71
DFHAIIN 69
DFHCPIN 69
DFHPRIN 70
DFHSIPLT 70
ECDSA 70, 90
EDF 70
EKCWAIT 70, 88
ERDSA 70, 90
ESDSA 70
EUDSA 70, 90
FCBFWAIT 70, 109
FCCIWAIT 71
FCDWWAIT 70, 110
FCFRWAIT 70
FCFSWAIT 70, 110
FCINWAIT 70
FCIOWAIT 70, 110
FCPSWAIT 70, 111
FCRAWAIT 111
FCRBWAIT 70
FCSRSUSP 70, 111
FCTISUSP 70, 111
FCXCWAIT 70, 112
FOREVER 71
HTYPE 53
ICEXPIRY 71
ICGTWAIT 71, 115
ICMIDNTE 71
ICWAIT 71, 115
inquiring during task waits 53
IRLINK 71, 94, 101
JASUBTAS 71, 85
JCAVLECB 71, 85
JCBUFFER 71, 85
JCCLDONE 71
JCDETACH 71
JCFLBUFF 71, 85
JCINITN 72
JCIOBLOK 72, 85
JCIOCOMP 72, 86
JCJASUS 72
JCLASTBK 72, 86
JCOPDONE 72, 86
JCREADY 72
JCRQDONE 72, 87
JCSWITCH 72, 87
JCTAPE2 72, 87
JCTERMN 73
resource types *(continued)*
JCTIMER 73
KC_ENQ 73, 88, 89, 106, 110
KCCOMPAT 73, 88, 94
KERNEL 73
MBCB_xxx 73, 107
MRCB_xxx 107
MXT 73
OPEN_ANY 73, 77
OPEN_NFY 73, 78
PROGRAM 73, 74
PS_INIT 74
RDSA 74
RECEIVE 74, 78
SDSA 74
STARTUP 74
SUBTASK 74
SUCNSOLE 74
summary of possible values 68
SUSPENDTYPE 53
TCP_NORM 74
TCP_SHUT 74
TCTVCECB 74
TD_INIT 74, 105
TDEPLOCK 74, 105
TDIPLOCK 75, 106
TDQ_IORB 75
TIEXPIRY 75
TRANDEF 75
TSAUX 75, 91
TSBUFFER 93
TSEXTEND 93
TSIO 93
TSMCPECB 75
TSQUEUE 93
TSSTRING 94
TSUT 94
TSWBUFFR 94
UDSA 75, 90
WAITLIST 75
WTOR_RPY 75
XOVR_RESP 75
XPCCWAIT 75, 78
XRF_SYNC 76
XRF_WAIT 76
XRGETMSG 76
XRPUTMSG 122
ZC 76, 103
ZC_ZCGRP 76, 103
ZC_ZGRP 76
ZC_ZGUB 76, 104
ZCIOWAIT 76, 104
ZCZGET 77, 104
ZCZNAC 77, 104
ZXQOWAIT 77, 104
ZXSTWAIT 77, 104

RESOURCE_NAME
ADD_SUSPEND input parameter 60
SUSPEND input parameter 62
WAIT_EXTERNAL input parameter 65
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
RESOURCE_TYPE
ADD_SUSPEND input parameter 60
SUSPEND input parameter 62
WAIT_EXTERNAL input parameter 65
WAIT_OLDC input parameter 65
WAIT_OLDW input parameter 65
resources
definition errors 5
inquiring during task waits 53
journal control 85
locks 82
investigating waits 83
names 68
storage manager 90
task control 88
temporary storage 91
types 68
RESPONSE
ADD_SUSPEND output parameter 60
DELETE_SUSPEND output parameter 62
INQUIRE_SUSPEND_TOKEN output parameter 61
meanings of possible values 67
RESUME output parameter 64
SUSPEND output parameter 63
WAIT_EXTERNAL output parameter 66
WAIT_OLDC output parameter 66
WAIT_OLDW output parameter 66
RESUME function 64
input parameters 64
RETAIN problem management system
APARs 265
data base 9, 262
problem management record 261
symptom keywords 9
using INFORMATION/ACCESS 9
runaway task time interval
See ICVR, system initialization parameter
runaway tasks
See tasks, runaway

S

SAA (storage accounting area)
chains 174
overlays 174, 177, 181
SAM 105
SCRNSIZE attribute, PROFILE 158, 162
SDAID, system debugging aid 23
SDSA resource type 74
SDUMPX macro 236
SENDSIZE attribute, TYPETERM 162
short on storage
See SOS (short on storage)
SHUTECB resource name 69
SINGLE resource name 73
SINGLE, resource name 70
SIPDMTEC resource name 69
SISUBECB resource name 74
SMSYSTEM resource name 74
SOS (short on storage)
caused by looping code 16
potential cause of waits 91
sources of information 19
SPCTR, system initialization parameter 215
SPCTRxx, system initialization parameter 215
SPECIAL_TYPE
SUSPEND input parameter 62
WAIT_OLDW input parameter 65
specification exception 32
spooler waits 77
STABLEnn resource name 72, 87
START PRINTER bit on write control character 165
STARTUP resource name 75
STARTUP resource type 74
statistics
autoinitiated tasks 16
file accesses 16
task control
number of times at MXT 142
use in investigating no task output 167
STGRCVY system initialization parameter 183
STNTR, system initialization parameter 215
STNTRxx, system initialization parameter 215
storage
consumption by looping tasks 91
fragmentation 91
task subpool summary 91
violations
See storage violations
waits 90
fragmentation of free storage 91
too little free storage 91
storage accounting area (SAA)
See SAA (storage accounting area)
storage chain checking
by CICS 174
forcing 178
storage freeze 252
storage manager domain (SM)
conditional storage requests 90
request for too much storage 90
suspend queue summary 90
trace levels 3 and 4 208
unconditional storage requests 90
waits 90
likely causes 90

storage recovery 183
storage violations
  CHKSTRM option 178
  CHKSTSK option 178
  CICS detected 173, 174
  CICS system dump 176, 177
  exception trace entry 176, 177, 180
  forcing storage chain checking 178
  investigating 173
  looking at the overlaying data 177
  possible causes 182
  programming errors 182
  reason for invalid ECB address 89
  symptoms 174
  TIOA 174
  undetected 173, 181
    possible causes 181
  user task storage element 174
  using CSFE DEBUG 178
  using trace 178, 181
STP_DONE resource name 69
STRFIELD option
  CONVERSE command 165
  SEND command 165
STRINGS parameter of FILE resource definition 111
structured fields 165
SUBTASK resource name 71, 72, 73, 86, 87
SUBTASK resource type 74
SUCNSOLE resource type 74
suppressing dumps
  CICS dump table options 154
SUSPEND function of gate DSSR 62
  input parameters 62
SUSPEND resource name 73
SUSPEND_TOKEN
  ADD_SUSPEND output parameter 60
  DELETE_SUSPEND input parameter 61
  INQUIRE_SUSPEND_TOKEN output parameter 61
  RESUME input parameter 64
  SUSPEND input parameter 62
SUSPENDTYPE field 53
SUSPENDVALUE field 53
symbolic maps 172
symptom strings
  in transaction dump 237
  problem determination 41
  RETAIN database search 237
  RETAIN search keywords 41
symptoms
  CICS has stopped running 10
  CICS running slowly 11
  incorrect output 13
  keywords 9, 261
  loops 14, 15
  low priority tasks will not start 11
  no output is obtained 11
symptoms *(continued)*
  poor performance 11, 14, 17, 139
  tasks do not complete 11
  tasks in a wait state 15
  tasks take a long time to complete 11
  tasks take a long time to start 11
  terminal activity is reduced 11
  use in classifying problems 10
  waits 14
SYSDUMP, system dump code attribute 155
SYSIDNT/session ID resource name 71
system busy symbol 16
system debugging aids (SDAIDS) 23
system dump
  formatting keywords and levels 245
system initialization
  AUXTR parameter 218
  AUXTRSW parameter 218
  defining component tracing requirements 215
  defining the tracing status 218
  DUMP parameter 154, 225
  DUMPDS parameter 236
  DUMPSW parameter 236
  global suppression of CICS system dumps 225
  INTTR parameter 218
  setting transaction dump data set attributes 235
  SPCTR parameter 215
  SPCTRxx parameter 215
  STNTR parameter 215
  STNTRxx parameter 215
  suppressing standard tracing 213
  SYSTR parameter 213
  TRTABSZ parameter 218
  USERTR parameter 218
system loading, effect on performance 146
system task waits 68, 122
  intentional 123
SYSTEM_DUMP, exit programming interface call 226
SYSTR, system initialization parameter 213

T

task control
  waits 88
    causes 88, 89
    failure of task to DEQUEUE on resource 89
    invalid ECB address 89
    resource type KCCOMPAT 88
    unconditional ENQUEUE on single server resource 89
    valid ECB address 89
task control area (TCA)
  in transaction dump 238
  system area 238
  user area 238

task global table (TGT) 242
task termination
  abnormal 3
task tracing
  precautions when choosing options 151
  special 211
  standard 211
  suppressed 211
tasks
  abnormal termination 3
  ATI, no output produced 165, 168
    looking at the AID chain 168
    looking at the ICE chain 168
  conversation state with terminal 100
  dispatch, suspend and resume cycle 141, 146
  dispatching priority 144
  error data 46
  exclusive control deadlock 112, 113
  failure to complete 11, 13, 14
  failure to get attached to the dispatcher 141, 142
  failure to get attached to the transaction manager 140, 141
  failure to get initial dispatch 141, 144
  failure to start 11, 12, 14
  identifying the AID 169
  identifying the ICE 169
  identifying, in remote region 101
  in a wait state 15
  in error 42
    identified in linkage stack 47
  information in kernel domain storage areas 44
  lock owning
    identifying a lock being waited on 82
  looping 129
    consumption of storage 91, 92
    identifying the limits 137
  MXT limit 142
  PSW at time of error 46
  reason for remaining on the AID chain 169
  registers at time of error 46
  runaway
    non-yielding loops 130
    storage report 44
    tight loops 130
  session state with VTAM 100
  slow running 12, 146
  subpool summary 91
  summary in kernel storage 42
  suspended 14
    inquiring on 15
    investigating 52
  task error information 44
  time-out interval 146
  tracing 151, 211
  transfer from ICE to AID chain 169
  waits 51
    definition of wait state 52
tasks *(continued)*
  waits *(continued)*
    journal control 85
    maximum task conditions 78
    on locked resources 82
    online investigation 52
    stages in resolving wait problems 52
    storage manager 90
    suspension and resumption of tasks 59
    system 68
    task control 88
    techniques for investigating 52
    temporary storage 91
    user 68
    using the formatted CICS system dump 52
    using trace 52
TCA (task control area)
  See task control area (TCA)
TCAPCDSA field 242
TCLASS resource type 74
TCP_NORM resource type 74
TCP_SHUT resource type 74
TCSESUSF 169
TCTCVECB resource name 76
TCTTE (terminal control table terminal entry)
  in transaction dump 240
TCTTE chain, in terminal waits 98
TCTVCEBC resource type 74
TCTVCECB resource name 69
TD_INIT resource type 74
TDEPLOCK resource type 74
TDIPLOCK resource type 75
TDQ_IORB resource type 75
temporary storage
  auxiliary control area 93
  byte map 93
  conditional requests for auxiliary storage 91
  consumption by looping tasks 92
  control interval size 93
  current free space 92
  no task output 168
  repetitive records 16
  request for too much storage 92
  suspend queue summary 92
  unconditional requests for auxiliary storage 91
  waits 91
    too little contiguous auxiliary storage 93
    unallocated space close to exhaustion 92
terminal control program (TCP) 98
terminal control table terminal entry (TCTTE)
  See TCTTE (terminal control table terminal entry)
TERMINAL resource name 73
terminal scan delay (ICVTSD)
  See ICVTSD, system initialization parameter
terminal tracing
  precautions when choosing options 151

terminal tracing *(continued)*
special 212
standard 212
suppressed 212
terminal waits
autoinstall program not loaded 95
CREATESESS(NO) in TYPETERM definition 99
error action by TACP or NACP turned off 95
finding the access method 96
HANDLE CONDITION coded incorrectly 95
interregion communication 101
intersystem communication 97, 101
multiregion operation 101
identifying tasks in remote regions 101
identifying the remote region 101
operator failing to respond 95
preliminary considerations 95
printer powered off 95
printer run out of paper 95
VTAM access method in use 97
automatic transaction initiation session status 99
NACP error codes 97
node session status 100
SNA sense codes 97
task conversation state with terminal 100
task session state with VTAM 100
task status 100
TCTTE chain 98
terminal control status 98
terminal status 98
VTAM exit ids 97
VTAM process status 97
VTAM terminal control 102
terminals
ATI status 169
control characters in data stream 158
conversation state with task 100
error messages 95
incorrect mapping of data 172
attributes of fields 172
DARK field attribute 172
MDT 172
modified data tag 172
symbolic map 172
incorrect output displayed
data formatted wrongly 162
debugging tools 163
early data overlaid by later data 162
investigating 156
logon rejection message 157
mixed English and Katakana characters 161
some data not displayed 162
unexpected messages and codes 158
unexpected uppercase or lowercase characters 158
wrong data values displayed 161
terminals *(continued)*
no input accepted 13
no output 11, 13
range of characteristics 95
reduced activity 11, 14, 16
repetitive output 16
status 98
effect on performance 141
terminal control program 98
unresponsive 94
waits
See terminal waits
termination
abnormal 3
system dump code option 229
termination stall 127
TIEXPIRY resource type 75
trace
abbreviated format 239
abbreviated-format 204
auxiliary
See auxiliary trace
calls to other programs 207
CETR transaction
See CETR transaction
CICS VTAM exit
destinations 149
CICS XRF trace 190, 221
description 221
destination 221
origin 221
controlling
auxiliary trace 218
internal trace 218
special task tracing 211
special trace levels 214
standard task tracing 211
standard trace levels 214
suppressing task tracing 211
data provided on call 207
exception trace 220
destinations 149
domain entry 207
domain exit 207
DSSR functions 54
input and output parameters 54
interpreting the trace table 54
entries from AP domain 200, 202
entries missing 151
example of formatted entry 201, 203
exception
See exception trace
extended-format 200, 239
formatted entry
data fields 201
interpretation string 200
interval 201

trace *(continued)*
formatted entry *(continued)*
kernel task number 201
standard information string 200
task number 200
time of entry 201
trace point id 200
formatting 199
from global trap/trace exit 254
global trap/trace exit 253
in problem determination
loops 134, 136
poor performance 143
principles 199
selecting destinations 216
storage violations 178, 181
incorrect output from
investigating 149
trace entries missing 151
wrong CICS components being traced 151
wrong data captured 150
wrong destination 149
wrong tasks being traced 151
internal
See internal trace
internal domain functions 207
interpreting 200, 204
user entries 206
interpreting user entries 206
investigating waits 52, 53
setting the tracing options 54
last command identification 241
last statement identification 242
level-1 207
level-2 208
level-3 208
levels 207, 220
logic of selectivity 213
master system trace flag 149, 213, 219
master user trace flag 219
overview of different types 199
point id 200
points 207
location 207
program check and abend 224
repetitive output 16
storage manager trace levels 208
suppressing standard tracing 213
suspension and resumption of tasks 54
interpreting the trace table 54
use in investigating no task output 166
user 171
checking programming logic 171
user exception trace entries 220
VTAM buffer
description 224
investigating logon rejection 157
trace *(continued)*
VTAM buffer *(continued)*
terminal waits 101
XRF
See trace, CICS XRF trace
TRANDEF resource type 75
TRANDUMP, transaction dump code attribute 155
transaction abends
abend code 27
documentation 28
interpretation 28
action of global trap/trace exit 254
AICA 29, 130
ASRA 29
ASRB 29
collecting the evidence 27
CSMT log messages 11
dump not made when expected 153
getting a transaction dump 27
investigating 27
last command identification 241
last statement identification 242
message 27
messages 3, 13
storage violation 176
system dump following 227
transaction dump following 227
worksheet 37
transaction dumps
abbreviated-format trace table 239
accompanying transaction abends 27
common system area 239
CSA 239
CSAOPFL 239
CWA 239
destination 235
DFHTACB 240, 242
dump not made on transaction abend 153
exec interface structure 238
EXEC interface user structure 239
execution key 238
extended-format trace table 239
following transaction abend 227
formatting 237
selectivity 237
in problem determination 225
interpretation 237
job log for DFHDU410 237
kernel stack entries 239
last command identification 241
last statement identification 242
locating program data 242
assembler program DFHEISTG storage 242
chain of PL/I DSAs 242
COBOL working storage 242
task global table 242

transaction dumps *(continued)*
module index 241
optional features list 239
program information 240
program storage 240
PSW 238
registers 238
registers at last EXEC command 238
remote abend indicator 238
selective dumping of storage 235
statistics 230
storage violation 176, 177
suppression for individual transactions 154, 225
symptom string 237
system EXEC interface block 238
task control area, system area 238
task control area, user area 238
TCTTE 240
transaction abend control block 240, 242
transaction dump code options 229
transaction dump codes 226
transaction storage 240
transaction work area 238
transaction ID resource name 75
transaction list table (XLT) 127
transaction manager
failure of tasks to get initial attach 141
transaction routing 101
transaction storage
in transaction dump 240
transaction tracing
precautions when choosing options 151
special 212
standard 212
suppressed 212
TRANSACTION_DUMP, exit programming interface
call 226
transactions
disabling 168
evidence that it ran 166
program control 166
program load 166
task attach 166
task dispatch 166
no output produced 164
ATI tasks 165, 168
disabling the transaction 168
explanatory messages 164
finding if the task ran 166
investigating 164
looking at files 168
looking at temporary storage queues 168
looking at transient data queues 168
possible causes 164
START PRINTER bit on write control
character 165
task not in system 165
transactions *(continued)*
no output produced *(continued)*
task still in system 165
testing the terminal status 164
using CECI 168
using execution diagnostic facility 166
using statistics 167, 168
using trace 166
wrong output produced 169
investigating 169
possible causes 169
transient data
extrapartition destinations 105
I/O buffers 107
intrapartition destinations 106
no task output 168
recoverable queues 106
VSAM I/O 107
VSAM strings 107
waits 105
during initialization 105
extrapartition 105
I/O buffer contention 107
I/O buffers all in use 107
intrapartition 106
resource names 105
resource types 105
VSAM I/O 107
VSAM strings all in use 107
transient data queue name resource name 75
translator
errors in output 6
traps 253
TRTABSZ, system initialization parameter 218
TSAUX resource type 75, 91
TSBMAP (temporary storage byte map) 93
TSBUFFER resource name 75, 93
TSEXTEND resource name 75, 93
TSIO resource name 75, 93
TSMCPECB resource name 74
TSMCPECB resource type 75
TSOPEN4B resource name 75
TSQUEUE resource name 75, 93
TSSTRING resource name 75, 94
TSUT resource name 75, 94
TSWBUFFR resource name 75, 94
TYPETERM definition
ALTPAGE attribute 158, 162
ALTSCREEN attribute 158, 162
ATI status 169
CREATESESS(NO), cause of terminal waits 99
EXTENDEDDS attribute 158, 162
PAGESIZE attribute 162
SENDSIZE attribute 162
UCTRAN attribute 158

U

UCTRAN attribute 160
  PROFILE definition 158
  TYPETERM definition 158
UDSA resource type 75
uppercase characters 158
user task waits 68
user tracing
  checking programming logic 171
  exception trace entries 220
  interpretation 206
USERTR, system initialization parameter 218

V

VARY NET,INACT command 128
VSAM
  data buffers 109
  exclusive control deadlock 112, 113
  exclusive control of control interval 112
  I/O waits 110
  incorrect data read from file 163
  index buffers 109
  strings 111
  transaction ID waits 111
  waits
    exclusive control deadlock 112, 113
    file state changes 110
    for exclusive control of control interval 112
    for VSAM transaction ID 111
    I/O 110
    record locking by CICS 114
    VSAM buffer unavailable 109
    VSAM string unavailable 111
VSAM READ SEQUENTIAL 113
VSAM READ UPDATE 113
VSAM WRITE DIRECT 113
VSAM WRITE SEQUENTIAL 113
VSE console
  CICS termination message 10
VSE dump library 236
VSE RESERVE locking ???????????
  CEDA in a wait state 122
  CESN in a wait state 122
  VSAM I/O waits 110
  waits during XRF takeover 122
  waits on resource type XRPUTMSG 122
VSE service aids
  EREP (Environmental Record Editing and Printing Program) 23
  SDAIDs, system debugging aids 23
VSE/ESA ABEND macro 29
VSMSTRNG resource name 69
VTAM
  buffer trace
    See trace, VTAM buffer
VTAM *(continued)*
  exit trace
    See trace, CICS VTAM exit
  process status 97
  session state with task 100
  terminal control waits 102

W

WAIT symptom keyword 9
WAIT_EXTERNAL function 64, 65
  input parameters 65
WAIT_OLDC function 64, 65
  input parameters 65
WAIT_OLDW function 64, 65
  input parameters 65
WAITLIST resource type 75
waits 77
  alternate system 121
  deadlock time-out interval 115
  definition 15, 52
  FEPI 123
  file control 108
  interregion communication 101, 104
  intersystem communication 101, 104
  interval control 115
  investigating 51
  journal control 85
  lock manager 82
  maximum task conditions 78
  online investigation 52
    finding the resource 53
  program control 108
  spooler waits 77
  stages in resolving 52
  storage manager 90
  suspension and resumption of tasks 59
  symptom keyword 9
  symptoms 14, 15
  task control 88
  techniques for investigating 52
  temporary storage 91
  terminal 94
  transient data 105
    during initialization 105
    extrapartition 105
    I/O buffer contention 107
    I/O buffers all in use 107
    intrapartition 106
    VSAM I/O 107
    VSAM strings all in use 107
  using the formatted CICS system dump 52, 54
  using trace 52, 53
    setting the tracing options 54
  VTAM terminal control 102

WCC (write control character) 165
WCSTECB resource name 75
working storage, COBOL programs 242
write control character (WCC) 165
WTO resource name 74
WTOR_RPY resource type 75

X

XDUREQ global user exit 225
XDUREQ, dump domain global user exit 154
XLT (transaction list table) 127
XM_HELD resource type 78
XOVR_RESP resource type 75
XPCCWAIT resource type 75
XPI
  See exit programming interface (XPI)
XRF errors
  active system does not initialize 185
  alternate system does not initialize 185
  alternate system terminated unexpectedly 188
  failure of CAVM 122
  investigating 185
  not all terminals recovered 187
  overseer did not restart failed job 189
  overseer sample program 189
    diagnostic messages 189
    return codes 189
    taking a snap dump 189
  related systems not taken over 188
  symptoms 185
  takeover did not occur when expected 187
  takeover does not complete 186
  unable to log on to CICS 189
  unexpected takeover 187
XRF takeover
  CEDA in a wait state 122
  CESN in a wait state 122
  wait on resource type XRPUTMSG 122
XRF trace
  See trace, CICS XRF trace
XRF_SYNC resource type 76
XRF_WAIT resource type 76
XRGETMSG resource type 76
XRPUTMSG resource type 76, 122
XRSSTECB resource name 76

Z

ZC resource type 76
ZC_ZCGRP 76
ZC_ZGRP 76
ZC_ZGRP resource name 74
ZC_ZGRP resource type 76
ZC_ZGUB 76
ZCIOWAIT 76
ZCZGET resource type 77
ZCZNAC resource type 77
ZSLSECB resource name 76
ZXQOWAIT resource type 77
ZXSTWAIT resource type 77

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

Problem Determination Guide

GC33-1663-00

If you want to send to IBM any comments you have about this book, please use one of the methods listed below. Feel free to comment on anything you regard as a specific error or omission in the subject matter, and on the clarity, organization or completeness of the book itself.

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.

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.

You can send your comments to IBM in any of the following ways:

- By mail:

  IBM UK Laboratories
  Information Development
  Mail Point 095
  Hursley Park
  Winchester, SO21 2JN
  England

- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Email: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5648-054

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC33-1663-00

Spine information:

CICS TS for VSE/ESA Problem Determination Guide *Release 1*