CICS Transaction Server for z/OS
5.4

Troubleshooting CICS

Note

Before using this information and the product it supports, read the information in "Notices" on page 295.

This edition applies to the IBM CICS® Transaction Server for z/OS® Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF

This PDF describes the facilities and methods for determining the cause of problems in a system that uses CICS. Other PDFs, listed below, describe how to address problems with certain areas of CICS and you might need to refer to those as well as this PDF. (In IBM Knowledge Center, all this information is under one section called "Troubleshooting".) Before CICS TS V5.4, this PDF was called *Problem Determination Guide*.

Troubleshooting information for areas of CICS is in the following PDFs:

- SOAP and JSON is in *Using Web Services with CICS* .
- ONC/RPC interface is in the *External Interfaces Guide* .
- EXCI is in *Using EXCI with CICS*.
- Java and Liberty are in *Java Applications in CICS*.
- Front End Programming Interface is in the *Front End Programming Interface User's Guide*.
- DB2 is in *Using Db2 with CICS*.
- DBCTL is in the *IMS DB Control Guide*.
- Shared data tables are in the *Shared Data Tables Guide*.
- CICSPlex SM is in *CICSPlex SM Administration*.
- BTS is in *Business Transaction Services*

For details of the terms and notation used in this book, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on 2024-01-04 (Year-Month-Date).

Chapter 1. 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.

About this task

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 might be useful to you later if you must carry out systematic problem determination.

Procedure

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 can check that CICS installed correctly by running batch or online verification procedures. For more information, see Verifying the CICS installation in Installing. If CICS did install successfully, check the appropriate Upgrading information set for any possible impacts to your system. If you are currently upgrading to, CICS Transaction Server for z/OS, Version 5 Release 4 ensure that you are aware of all the changes that were made for this release. For details, see What's New and Upgrading for the release from which you are upgrading.
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 startup, or while the system was running before the error occurred? These messages might indicate some system problem that prevented your transaction from running successfully. If you see any messages that you do not understand, use the CICS messages transaction, CMAC, for online message information. If you do not have access to a CICS system to run the CMAC transaction, look in CICS messages for an explanation. A suggested course of action that you can take to resolve the problem might also be included with the explanation.
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.
   - Check that you sufficient resources are defined for the application, such as VSAM file strings. Typically, if the resources defined are not sufficient, 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 user exit programming interface (XPI). The exit programming interface is used to start 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 can cause unpredictable effects.
   - 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. See Global user exit programs for details on which exits can use XPI calls and which cannot.

   c) Consider your CICS system definition parameters if the problem is not related to any particular application.

   Poorly defined parameters can be the cause of problems in your system. You can find guidance about setting up your CICS system in, Specifying CICS system initialization parameters

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 problem, see Improving the performance of a CICS system.

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

If the failure occurs at specific times of day, it can be dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so those times are when load-dependent failures are most likely to happen. If your CICS network extends across more than one time zone, peak system loading might seem to you to 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 be deleted from the system long before the symptoms are seen. A method you can use to investigate random overlays is described in. "Dealing with storage violations" on page 231

6. **Have you made any service changes since the last successful run?**

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?

7. **Have you made any hardware changes since the last successful run?**
8. **Have you made any software changes since the last successful run?**

If you installed a new or modified application, check for error messages in the output from the:

- Translator
- Compiler
- Assembler
- Linkage editor

9. **Have you made any administrative changes since the last successful run?**

a) Have you changed your initialization procedure, for example by JCL, CICS system initialization or override parameters, or z/OS Communications Server 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 be preserved over the termination and subsequent startup. 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 the GRPLIST specified on a *cold* start, it is effectively installed during startup. (Changes which were installed are not visible after a cold start unless they were made to a group in the GRPLIST.)If **START=AUTO** was specified in the system initialization table, or as an override, you must examine the job log to find out how the CICS system last came up.

d) Have you changed the configuration or the status of installed resources in CICS?

The CICS configuration might have changed. Installed resources might be disabled or closed. For example, a file or program might be disabled. These changes are made using CEMT transactions, the Web User Interface, CICS explorer, or CICS applications. CICS provides a function to audit these changes by writing a message to the job log when these commands are issued. The message DFHAP1900 contains the date, time, transaction id, netname, user ID, and detail of the command. A system administrator, or anyone wanting to review audit records, can read the audit messages in the CICS job log to find any changes that were made to this CICS region. For commands using generic parameters, each individual command is audited, so a search for a specific file name, for example, would succeed in finding that file.

10. **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 was 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 TERMINAL definition, and the 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 TYPETERM definition? If so, it is likely that there is an error in that TYPETERM definition.
- Is the whole network affected? If so, CICS might have stalled. See "What to do if CICS has stalled" on page 162 for advice about dealing with CICS system stalls.

11. **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 map set? 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
```

.

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 ran successfully on many previous occasions, examine the contents of any records, screen data, and files that were being processed when the error occurred. They might contain some unusual data value that causes 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.

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

If your application has not yet run successfully, examine it carefully for any errors.

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.

Chapter 2. Classifying the problem

The purpose of this section is to help you classify your problem into one of the categories used by the IBM Support Center for its service procedures. IBM Support Center staff have found that classifying the problem first is a good approach to problem determination.

It contains the following topics:

Using symptom keywords to classify problems

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.

About this task

If you have the IBM INFORMATION/ACCESS licensed program, 5665-266, you can look on the RETAIN database yourself. Each problem in the database has a classification type.

Procedure

- Classify your problem using one of the following software categories from RETAIN.

  Use the appropriate reference to get further information on how to diagnose each category of problem.

  - **ABEND** (for **transaction abends**, see "Dealing with transaction abend codes" on page 87; for **system abends**, see "Dealing with CICS system abends" on page 97)
  - **WAIT** (see "Dealing with waits" on page 110)
  - **LOOP** (see "Dealing with loops" on page 196)
  - **POOR PERFORMANCE**, or **PERFM** (see "Dealing with performance problems" on page 204)
  - **INCORRECT OUTPUT**, or **INCORROUT** (see "Dealing with incorrect output" on page 210)
  - **MESSAGE**

  All theses categories are considered in the information on problem determination, except for the MESSAGE category. If a CICS error message is issued, you can use the CICS message transaction, CMAC, for online message information. See CMAC - messages and codes display. If you do not have access to a running CICS system, see CICS messages for an explanation. If you get a message from another IBM program, or from the operating system, see the messages and codes information for the appropriate product for an explanation of that message.

  CICS messages might provide enough information to solve the problem quickly, or it might redirect you to other information sources for further guidance. If you cannot deal with the message, you might eventually need to contact the IBM Support Center for help.

  One type of problem that might result in 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 information. However, for an example of how poor design can result in application problems, see "Poor application design" on page 13.

Using the symptoms to classify the problem

You can 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 might 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.

About this task

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

CICS has stopped running

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.

Consider, too, 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 7 to confirm this before going on to "Dealing with performance problems" on page 204 for advice about what to do next.

If CICS *has* stopped running, look for any message that might explain the situation. The message might appear in either of the following places:

- **The MVS™ 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 and that CICS is no longer running. In such a case, you need to examine the CSMT log (see below) to see which abend message has been written there.

  If you do not find any explanatory message on the MVS console, check in the CSMT log to see if anything has been written there.
- **The CSMT log**. CSMT is the transient data destination to which abend messages are written. If you find a message there, use the CMAC transaction or look in CICS messages 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 if 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 "Dealing with CICS system abends" on page 97 for advice on what to do next.

If you can find no message saying 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 "Dealing with waits" on page 110 and "Dealing with loops" on page 196, 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 seem to you to 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. You might classify the problem initially as "poor performance", but be prepared to reconsider your classification later.

The following are some individual 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, seeWhat to investigate when analyzing performance.

You can find guidance about identifying specific performance bottlenecks in your CICS system in "Dealing with performance problems" on page 204.

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, the transaction is queuing for admittance to a transaction class, or for other performance reasons.

Classify the problem tentatively as "poor performance", and turn to "Dealing with performance problems" on page 204 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 Chapter 3, "Distinguishing between waits, loops, and poor performance," on page 11.

Note: Do not overlook the possibility that the task might 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 **CEMT INQ TASK** to check its status, and make sure that it has not ended without 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:

- CDBC, the destination for DBCTL related messages
- 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 Setting up data sets for transient data. 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 Chapter 3, “Distinguishing between waits, loops, and poor performance,” on page 11 to find out which of these is the most likely explanation. You can then read to the appropriate section 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.

Use the CMAC transaction or look in CICS messages 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 “Dealing with transaction abend codes” on page 87.

You have obtained some incorrect output

Incorrect output might be regarded as 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, use the CMAC transaction or look in CICS messages for an explanation. It might suggest a simple response that you can make to the message, or it might direct you to other sources of information for further guidance.

These are the types of incorrect output that are dealt with in this information:

- **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**.

You can find advice about investigating the cause of any of these types of incorrect output in "Dealing with incorrect output" on page 210.

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 "Dealing with storage violations" on page 231 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 transaction isolation, storage protection, and command protection.

Chapter 3. 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.

About this task

Any of the following symptoms could be caused by a wait, a loop, 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. However, note that in some cases your initial classification could be wrong, and you will then need to reappraise the problem.

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 ECB to be posted. A wait might affect just a single task, or a group of tasks that may be related in some way. 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 162.

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 CEMT INQ TASK. 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 as alternatives to the CEMT transaction. You can execute these commands under CECI, or in a user program.

Use INQUIRE TASK LIST 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 “Dealing with waits” on page 110 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 MVS job by using the DISPLAY ACTIVE command at the MVS console to display the active users.
- **There is reduced activity at terminals**, or possibly no activity at all.
- **One or more CICS regions 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 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 "Dealing with loops" on page 196.

If you are able to use the CEMT transaction, try issuing CEMT INQ TASK 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 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 162.

Consider the evidence you have so far. Does it indicate a loop? If so, turn to "Dealing with loops" on page 196, where there are procedures for defining the limits of the loop.

Poor performance

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

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

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

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

  CICS is under stress in one of the dynamic storage areas (DSAs) in 24-bit storage.
- DFHSM0133 *applid* CICS is under stress (short on storage above 16 MB)

  CICS is under stress in one of the DSAs in 31-bit storage.
- DFHSM0606 *applid* The amount of MVS above-the-bar storage available to CICS is critically low.

  CICS is under stress in 64-bit storage.

You can also use CICS storage manager statistics to identify the situation. See Short-on-storage conditions in dynamic storage areas.

If there is no such indication, see "Dealing with performance problems" on page 204 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, or a wait, or possibly a performance problem with an individual transaction, consider the possibility that poor design might be to blame.

An example of poor application design is given here, to show how this can give 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 might 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.

Chapter 4. Classifying problems by functional area

About this task

In addition to the RETAIN classifications used by the IBM Support Centers, the following types of problem also belong in classes of their own:

- EXCI problems - see “Dealing with external CICS interface (EXCI) problems” on page 238
- MRO problems
- Log manager problems - see “Dealing with log manager problems” on page 241
- Java™ problems - see Troubleshooting Java applications
- Storage violations - see “Dealing with storage violations” on page 231

Whereas 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 “Using symptom keywords to classify problems” on page 5.

Chapter 5. Diagnostic tools and information

You should find some of the following sources of information useful in problem determination.

Documentation

"Your own documentation" is the collection of information produced by your organization about what your system and applications do and how they do it. Product documentation is supplied by IBM.

About this task

How much of this kind of information you need depends on how familiar you are with the system or application, and could include:

- Program descriptions or functional specifications
- Record layouts and file descriptions
- Flowcharts or other descriptions of the flow of activity in a system
- 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.

Source listings and link-edit maps

Include the source listings of any applications written at your installation with your set of documentation. They often form the largest single element of documentation. Large installations with thousands of programs might prefer to keep such listings on 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 a load module 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 load library in which the load module was placed.

Abend codes and error messages

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

- CDBC for messages concerning DBCTL
- CSFL for file control messages.

For a list of the destinations used by CICS, see Setting up data sets for transient data. Use a copy of the appropriate messages and codes documentation to look up any messages whose meaning you do not know. All CICS messages and codes are documented in CICS messages. Make sure that you also have some documentation of application messages and codes for programs that were written at your installation.

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 directly typed in and 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 tell you of 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 MVS and CICS) changes, application changes, and any modifications made to operating procedures.

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 region.

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 than the transaction dump. You can be reasonably confident that the system dump has captured all the evidence you need to solve your problem, but it is possible that the transaction dump might have missed some important information.

The amount of CICS system dump data that you could get is potentially very large, but that 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. There are two aspects to controlling dump action:

1. Setting up the dumping environment, so that the appropriate dump action is taken when circumstances arise that might cause a dump to be taken.
2. Causing a dump to be taken. Both users and CICS can issue requests for dumps to be taken.

For information about using dumps to solve FEPI problems, see FEPI dump.

Setting up the dumping environment

There are several levels at which the dumping environment can be set up.

About this task

Procedure

- To globally suppress or enable system dumps at system initialization, use the **DUMP** system initialization parameter

  This does not apply to CICS kernel domain dumps.
- To globally suppress or enable system dumps dynamically, use the **EXEC CICS SET SYSTEM DUMPING** command.

  This does not apply to CICS kernel domain dumps.
- To enable or suppress transaction dumps for individual transactions, use one of the following options:
  - **EXEC CICS SET TRANSACTION DUMPING** system programming command
  - **CEMT SET TRDUMPCODE** command
  - DUMP attribute of the RDO definition for the transaction.
- To suppress specific dump codes from a dump domain, use the XDUREQ global user exit program.

  This does not apply to CICS kernel domain dumps.
- To suppress or enable system dumps (apart from CICS kernel domain dumps), and specifying other dumping requirements, use dump codes.

  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 30.
- To enable dumps of the builder parameter at specific stages in the build process of terminal or connection definitions, use the CSFE ZCQTRACE facility.

  For details, see "The CSFE ZCQTRACE facility" on page 37.

Detecting and avoiding duplicate system dumps

When more than one CICS system runs under one instance of the MVS operating system, two CICS systems can take duplicate system dumps.

Each CICS system dump header includes a symptom string. The symptom string will be created only if the system dump code has the DAE option specified in the dump table entry. The default action is that symptom strings are not produced. This can, however, be altered by means of the **DAE** system initialization parameter.

The symptom strings provide sufficient information to enable the detection of duplicate dumps. You can take advantage of this in either of two ways:

1. Use MVS Dump Analysis Elimination (DAE) to detect and suppress duplicate dumps. (If the symptom string has been suppressed by the dump table option, DAE will not suppress the system dump.)

   You can control DAE with an ADYSETxx parmlib member. For information about DAE, see z/OS MVS Diagnosis: Tools and Service Aids.
2. Manually compare the headers of system dumps, so that you are aware that you have duplicate dumps. Doing it this way, you avoid repeating the same analysis, but still have a separate dump listing for each CICS system.

Where dumps are written

Transaction dumps and system dumps are written to different destinations.

- **Transaction dumps** go to a pair of CICS BSAM data sets, with DD 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 INQ 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** 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 overflowing transaction dump is written in its entirety 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 MVS dump data sets. CICS can write only one system dump to any individual dump data set.

  If another address space is already taking an SDUMP when CICS issues the SDUMP macro, the request fails but is automatically retried. You can use the **DURETRY** system initialization parameter to define the total time that CICS is to continue trying to take an SDUMP.

  If CICS tries to take an SDUMP when all the dump data sets are full, the dump is lost. Because of this, it is advisable to monitor the number of full data sets, and to take copies of dumps before processing them, so that you can free the dump data sets for reuse.

Events that can cause dumps to be taken

The following are the events that can 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.

Note: Due to the circumstances under which they are called, the following abends might not always produce a transaction dump:

ASPF
ASPN
ASPO
ASPP
ASPQ
ASPR
ASP1
ASP2
ASP3
ASP7

ASP8

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 main 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 â€œMT0001â€.
- 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 an exit program to get a transaction dump or a system dump, respectively. For programming information about these exits, see Global user exit programs.

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.

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 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 taken.

CICS does not take a transaction dump if a HANDLE ABEND is active at the current logical level. This is called an implicit HANDLE ABEND and causes the suppression of transaction dumps. To make PL/I work on units, PL/I library routines can issue HANDLE ABEND. 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 CICS messages. 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 CICS messages.

- When a transaction dump or system dump is taken, and the dump code includes the RELATED attribute on the DUMPSCOPE option, system dumps are taken of all CICS regions in the sysplex which are related to the CICS region on which this transaction dump or system dump is taken. A related CICS region is one in which the unit of work identifiers, in the form of APPC tokens, of one or more tasks match those in the CICS region that issued the dump request. Typically, these may be regions in a distributed transaction processing environment.
- 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 message DFHZC3496 and the the programming information on NEPs in Writing a node error program.
- A CICS system dump can also be requested from the global trap/trace exit. In this case, the system dump code is TR1003.

CICS dumps in a sysplex

You can capture simultaneous dump data from multiple CICS regions across a sysplex. This facility helps with problem determination in XCF/MRO environments where many CICS regions are running.

Capturing dump data in this way is useful in the following situations:

- A task involves multiple CICS regions in a sysplex and one region issues a dump, typically in response to an error.

  To fully diagnose and solve the problem, dump data from all CICS regions related to the region that issues the dump request is usually required. This dump data must be captured at the same time as the dump taken on the region that issues the dump.
- A MVS console operator needs to capture, simultaneously, dump data from multiple CICS regions in the sysplex.

To collect dump data in this way, you need MVS/ESA 5.1, the z/OS Workload Manager (WLM), and the XCF facility. The z/OS images in the sysplex must be connected through XCF. The CICS regions must be using MRO that is supported by the CICS TS 5.4 interregion communication program, DFHIRP.

Automatic dump data capture from related CICS regions

It is possible to collect dump data simultaneously from all related CICS regions in a sysplex. Related CICS regions are those containing one or more tasks which have unit of work identifiers, in the form of APPC tokens, that match the unit of work identifiers in the CICS region which initially issued the dump request.

The CICS regions must be connected via XCF/MRO. Connections using thez/OS Communications Server (SNA) ISC are not eligible to use the related dump facility.

The function is controlled by the DUMPSCOPE option on each CICS dump table entry. You can set this option to have either of the following values:

- RELATED - take dumps for all related CICS regions across the sysplex.
- LOCAL - take dumps for the requesting CICS region only. This is the default.

The DUMPSCOPE option is available on the following main terminal and system programming commands:

- **EXEC CICS INQUIRE SYSDUMPCODE**
- **EXEC CICS SET SYSDUMPCODE**
- **EXEC CICS INQUIRE TRANDUMPCODE**
- **EXEC CICS SET TRANDUMPCODE**
- **CEMT INQUIRE SYDUMPCODE**

- **CEMT SET SYDUMPCODE**
- **CEMT INQUIRE TRDUMPCODE**
- **CEMT SET TRDUMPCODE**

If the DUMPSCOPE option is set to RELATED in the CICS region issuing the dump request, a request for a system dump is sent to all MVS images in the sysplex that run related CICS regions.

The local MVS image running the CICS region that initiated the dump request has two dumps - one of the originating CICS region, the other containing the originating CICS region and up to fourteen additional related CICS regions from the local MVS image.

When a dump is requested, the DUMPSCOPE option is tested only in the CICS region issuing the original dump request. If the requesting CICS region has DUMPSCOPE defined as RELATED for the dump code, then all related CICS regions are dumped even if they have DUMPSCOPE defined as LOCAL for the dump code.

There is a maximum of fifteen address spaces in an SDUMP. If there are more than fifteen related CICS regions on an MVS image, then not all of them will be dumped. Related CICS regions may also fail to be dumped if they are swapped out when the dump request is issued. You should consider whether to make certain CICS regions non-swappable as a result.

Operator-requested simultaneous dump data capture

The MVS console operator might want to issue a dump request simultaneously to several CICS regions in the sysplex. There might be a problem in the sysplex where one or more CICS regions is hanging and, to fully diagnose and solve the problem, dump data captured at the same time is required.

Without this facility, such simultaneous dump data capture across multiple CICS regions in the sysplex is impossible.

Use the following command at the console:

```
DUMP COMM=()
R x,REMOTE=(SYSLIST=*),PROBDESC=(SYSDCOND,SYSDLOCL,(DFHJOBN,jobnames))
```

where:

- REMOTE controls the issuing of dumps on remote systems.
- SYSLIST=* means the request is to be routed to all remote systems.
- PROBDESC is problem description information, as follows:
  - SYSDCOND is an MVS keyword that specifies that a dump is to be taken on remote MVS images if the IEASDUMP.QUERY exit responds with return code 0. CICS supplies DFHDUMPX as the IEASDUMP.QUERY exit.
  - SYSDLOCL is an MVS keyword that drives the IEASDUMP.QUERY exit on the local and remote MVS images. This allows the CICS regions on the local MVS region to be dumped.
  - DFHJOBN is a CICS keyword. The operator should include the generic job name. This is used by DFHDUMPX to determine which address spaces to dump.

For a description of all command options, see z/OS MVS System Commands.

If you adopt a suitable naming convention for your CICS regions, this can be used to define suitable generic job names to determine which CICS regions to dump. For recommendations on naming conventions, see z/OS V1R1.0-V1R12.0 Parallel Sysplex Application Migration. If you follow these recommendations, the generic job name for all CICS regions in the sysplex would be CICS*.

Requesting dumps to resolve SMSVSAM problems

You may sometimes encounter a CICS problem that involves SMSVSAM. If you need to submit such a problem to IBM, in particular where CICS or a CICS transaction is in a hung state while accessing VSAM data sets in RLS mode, include a dump of the following items.

- All the SMSVSAM server address spaces in the sysplex, together with their associated data spaces
- The address space of the CICS region that is hanging
- The GRS address space
- The VSAM CATALOG address space.

You can obtain such a dump by using the following command:

```
/DUMP COMM=(comment to describe the problem)
```

When MVS responds, reply to WTOR number *nn* with the following:

```
/R nn,JOBNAME=(CICS-job-name,SMSVSAM,CATALOG,GRS),DSPNAME='SMSVSAM'.*,
      REMOTE=(SYSLIST=*('SMSVSAM','CATALOG','GRS'),DSPNAME,SDATA,
      END
```

Note: You can use the CONT option to split this command into parts, as follows:

```
/R nn,JOBNAME=(CICS-job-name,SMSVSAM,CATALOG,GRS), CONT
/R nn,DSPNAME='SMSVSAM'.*,REMOTE=(SYSLIST=*('SMSVSAM', CONT
/R nn,'CATALOG','GRS'),DSPNAME,SDATA, END
```

Useful CICS main terminal and MVS console commands in a sysplex

MVS support for remote SDUMPs is available only for images running MVS/ESA 5.1 or later and which are connected by XCF. Related CICS SDUMPs are produced only for CICS regions which are MRO connected using XCF.

DFHIRP must be at CICS TS 5.4 level. Connections using z/OS Communications Server ISC are not eligible to use the related dump facility.

If you are unable to produce related system dumps when there are related CICS regions across MVS images, ensure that the regions are MRO connected.

Use the command **CEMT I IRC** to ensure that interregion communication is available. If IRC is not available it may be started using the **CEMT S IRC OPEN** command. Failure to start IRC results in DFHIRxxx messages which may be used to identify the source of the problem.

During IRC start processing, CICS attempts to join XCF group DFHIR000. If this fails, return code yyy is given in the DFHIR3777 message.

The following MVS console commands may be used to monitor activity in the sysplex:

- D XCF - to identify the MVS sysplex and list the name of each MVS image in the sysplex.

  An example of a response to this command looks like this:

  ```
  08.14.16 DEV5           d xcf
  08.14.16 DEV5           IXC334I  08.14.16  DISPLAY XCF 602
     SYSPLEX DEVPLEX5:   DEV5
  ```

  This response tells you that MVS image DEV5 has joined sysplex DEVPLEX5.
- D XCF,GROUP - to list active XCF groups by name and size (note that CICS, group DFHIR000, is missing from the following example response).

```
16.21.36 DEV5           d xcf,group
16.21.36 DEV5           IXC331I  16.21.36  DISPLAY XCF 877
   GROUPS(SIZE):   COFVLFNO(1)       SYSDAE(2)         SYSGRS(1)
                   SYSIGW00(1)       SYSIGW01(1)       SYSIKJBC(1)
                   SYSMCS(6)         SYSMCS2(3)        SYSWLM(1)
                   WINMVSG(1)
```

- D XCF,COUPLE - to list details about the XCF coupling data set and its definitions. In the following example response, the data set has a MAXGROUP of 10 and a peak of 10. The response to XCF,GROUP indicates there are currently 10 active groups. If CICS now attempts to join XCF group DFHIR000, it will be rejected and the IRC start will fail with message DFHIR3777.

```
16.30.45 DEV5           d xcf,couple
16.30.45 DEV5           IXC357I  16.30.45  DISPLAY XCF 883
SYSTEM DEV5 DATA
   INTERVAL   OPNOTIFY     MAXMSG    CLEANUP      RETRY   CLASSLEN
         42         45        500         60         10        956

   SSUM ACTION      SSUM INTERVAL             WEIGHT
          N/A                N/A                N/A
SYSPLEX COUPLE DATA SETS
PRIMARY    DSN: SYS1.XCFDEV5.PXCF
           VOLSER: SYS001     DEVN: 0F0E
           FORMAT TOD         MAXSYSTEM MAXGROUP(PEAK) MAXMEMBER(PEAK)
           08/16/93 08:05:39          8      10   (10)       87    (6)
ALTERNATE  DSN: SYS1.XCFDEV5.AXCF
           VOLSER: SYS001     DEVN: 0F0E
           FORMAT TOD         MAXSYSTEM    MAXGROUP        MAXMEMBER
           08/16/93 08:05:41          8       10             87
```

This example also indicates that the primary and alternate data sets are on the same volume, thereby giving rise to a single point of failure.

Use the command **CEMT I CONNECTION** to display the status of the connections. ‘XCF' is displayed for every acquired connection using MRO/XCF for communications.

```
  I CONNECTION
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Con(FORD) Net(IYAHZCES)     Ins Acq         Xcf
   Con(F100) Net(IYAHZCEC)     Ins Acq         Irc
   Con(F150) Net(IYAHZCED)     Ins Acq         Irc
   Con(GEO ) Net(IYAHZCEG)     Ins Acq         Xcf
   Con(GMC ) Net(IYAHZCEB)     Ins Acq         Xcf
   Con(JIM ) Net(IYAHZCEJ)     Ins Acq         Xcf
   Con(MARY) Net(IYAHZCEM)     Ins Acq         Xcf
   Con(MIKE) Net(IYAHZCEI)     Ins Acq         Xcf
 + Con(RAMB) Net(IYAHZCEE)     Ins Acq         Xcf
                                                  SYSID=CHEV APPLID=IYAHZCET
   RESPONSE: NORMAL                            TIME:  01.28.59  DATE: 06.11.94
 PF 1 HELP       3 END                    7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

CICS initialization issues an MVS CSVDYNEX request to add DFHDUMPX as an MVS IEASDUMP.QUERY exit. If you need to change the status of the exit, use the MVS console command SETPROG EXIT.

If you have determined that XCF communication is in use, you can verify that the CICS SDUMP exit has been established using the following MVS commands:

```
D PROG,EXIT,MODNAME=DFHDUMPX
    08.16.04 DEV5          CSV463I MODULE DFHDUMPX IS NOT ASSOCIATED ANY EXIT

D PROG,EXIT,EN=IEASDUMP.QUERY
   08.17.44 DEV5           CSV463I NO MODULES ARE ASSOCIATED WITH EXIT IEASDUMP.QUERY
```

This example indicates that the exit has not been established.

The following displays indicate that DFHDUMPX is active as an IEASDUMP exit with one CICS TS 5.4 region active in one MVS image.

```
D PROG,EXIT,MODNAME=DFHDUMPX
01.19.16 DEV5           CSV461I 01.19.16 PROG,EXIT DISPLAY 993
EXIT             MODULE   STATE MODULE   STATE MODULE   STATE
IEASDUMP.QUERY   DFHDUMPX   A

D PROG,EXIT,EN=IEASDUMP.QUERY
01.19.46 DEV5           CSV462I 01.19.46 PROG,EXIT DISPLAY 996
MODULE  DFHDUMPX
EXIT(S) IEASDUMP.QUERY
```

You may issue MVS dump commands from the console to verify that remote dumping is available within the MVS image, without an active CICS region.

```
11.29.59 DEV5           dump comm=(NO CICS)
11.29.59 DEV5          *03 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
11.36.49 DEV5           r 03,remote=(syslist=*),probdesc=(sysdcond,
 sysdlocl,(dfhjobn,iyahzcet))
11.36.49 DEV5           IEE600I REPLY TO 03 IS;REMOTE=(SYSLIST=*),
 PROBDESC=(SYSDCOND,SYSDL
11.36.52 DEV5           IEA794I SVC DUMP HAS CAPTURED:
DUMPID=001 REQUESTED BY JOB (*MAIN*)
DUMP TITLE=NO CICS
```

In the next example, the messages from SDUMP indicate that one dump of the main address space has been taken.

```
*11.37.03 DEV5          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
*DUMPID=001 REQUESTED BY JOB (*MAIN*)
*FOR ASID (0001)
*REMOTE DUMPS REQUESTED
*INCIDENT TOKEN: DEVPLEX5 DEV5     06/28/1994 11:36:49
```

Another test is to issue the dump command specifying the CICS XCF group.

```
11.42.33 DEV5           dump comm=(STILL NO CICS)
11.42.33 DEV5          *05 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
11.43.27 DEV5           r 05,remote=(grplist=dfhir000(*)),
 probdesc=(sysdcond,sysdlocl,(dfhjobn,iyahzcet))
11.43.28 DEV5           IEE600I REPLY TO 05
 IS;REMOTE=(GRPLIST=DFHIR000(*)),PROBDESC=(SYSD
11.43.31 DEV5           IEA794I SVC DUMP HAS CAPTURED:
DUMPID=002 REQUESTED BY JOB (*MAIN*)
DUMP TITLE=STILL NO CICS
```

The messages from SDUMP indicate that one dump of the main address space has been taken.

```
*11.43.42 DEV5          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
*DUMPID=002 REQUESTED BY JOB (*MAIN*)
*FOR ASID (0001)
*REMOTE DUMPS REQUESTED
*INCIDENT TOKEN: DEVPLEX5 DEV5     06/28/1994 11:43:28
```

To verify that the remote dumping function works on the local system, use the following commands:

```
11.45.57 DEV5           dump comm=(TEST REMOTE FUNCTION ON LOCAL SYSTEM
11.45.57 DEV5          *06 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
11.46.57 DEV5           r 06,remote=(grplist=*(*)),probdesc=(sysdlocl)
11.46.59 DEV5           IEE600I REPLY TO 06 IS;REMOTE=(GRPLIST=*(*)),
 PROBDESC=(SYSDLOCL)
11.47.00 DEV5           IEA794I SVC DUMP HAS CAPTURED:
DUMPID=003 REQUESTED BY JOB (*MAIN*)
DUMP TITLE=TEST REMOTE FUNCTION ON LOCAL SYSTEM
11.47.17 DEV5           IEA794I SVC DUMP HAS CAPTURED:
DUMPID=004 REQUESTED BY JOB (DUMPSRV )
DUMP TITLE=TEST REMOTE FUNCTION ON LOCAL SYSTEM
```

The messages from SDUMP indicate two dumps were taken, one for the main address space and a second which contains ASIDs 0101, 0012, 0001, 0005, 000B, 000A, 0008, 0007. Note that the same incident token is used for both dumps.

```
*11.47.39 DEV5          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
*DUMPID=003 REQUESTED BY JOB (*MAIN*)
*FOR ASID (0001)
*REMOTE DUMPS REQUESTED
*INCIDENT TOKEN: DEVPLEX5 DEV5     06/28/1994 11:46:57
*11.47.59 DEV5          *IEA911E COMPLETE DUMP ON SYS1.DUMP04
*DUMPID=004 REQUESTED BY JOB (DUMPSRV )
*FOR ASIDS(0101,0012,0005,0001,000A,000B,0008,0007)
*REMOTE DUMP FOR SYSNAME: DEV5
*INCIDENT TOKEN: DEVPLEX5 DEV5     06/28/1994 11:46:57
```

The following example lists the MVS console messages received when the CICS main terminal command CEMT P DUMP is issued from CICS APPLID IYAHZCET executing in ASID 19 on MVS image DEV6. IYAHZCET has at least one related task in the CICS region executing in ASID 1B on MVS DEV6 and ASIDS 001A, 001C, 001B, 001E, 001F, 0020, 001D, 0022, 0024, 0021, 0023, 0028, 0025, 0029, 0026 on MVS image DEV7.

```
- 22.19.16 DEV6 JOB00029  +DFHDU0201 IYAHZCET ABOUT TO TAKE SDUMP. DUMPCODE: MT0001
- 22.19.23 DEV7           DFHDU0214 DFHDUMPX IS ABOUT TO REQUEST A REMOTE SDUMPX.
- 22.19.23 DEV6           DFHDU0214 DFHDUMPX IS ABOUT TO REQUEST A REMOTE SDUMPX.
  22.19.27 DEV6 JOB00029  IEA794I SVC DUMP HAS CAPTURED:
  DUMPID=001 REQUESTED BY JOB (IYAHZCET)
  DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCET CODE=MT0001   ID=1/0001

  22.19.43 DEV6 JOB00029  IEA794I SVC DUMP HAS CAPTURED:
  DUMPID=002 REQUESTED BY JOB (DUMPSRV )
  DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCET CODE=MT0001   ID=1/0001

- 22.19.43 DEV6 JOB00029  +DFHDU0202 IYAHZCET SDUMPX COMPLETE. SDUMPX RETURN CODE X'00'

 *22.21.00 DEV6          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
 *DUMPID=001 REQUESTED BY JOB (IYAHZCET)
 *FOR ASID (0019)
 *REMOTE DUMPS REQUESTED
 *INCIDENT TOKEN: DEVPLEX1 DEV6     06/10/1994 22:19:16
 *ID = DUMP : APPLID IYAHZCET DUMPCODE MT0001  /1/0001
```

The dump in SYS1.DUMP03 on DEV6 was taken as a result of the CEMT request on IYAHZCET.

```
*22.21.15 DEV6          *IEA911E COMPLETE DUMP ON SYS1.DUMP04
*DUMPID=002 REQUESTED BY JOB (DUMPSRV )
*FOR ASIDS(0019,001B)
*REMOTE DUMP FOR SYSNAME: DEV6
*INCIDENT TOKEN: DEVPLEX1 DEV6     06/10/1994 22:19:16
*ID = DUMP : APPLID IYAHZCET DUMPCODE MT0001  /1/0001
```

The dump in SYS1.DUMP04 on DEV6 was taken as a remote dump by MVS dump services as a result of the CEMT request on IYAHZCET. Note that the incident token and ID are the same.

```
 22.22.35 DEV7 JOB00088  IEA794I SVC DUMP HAS CAPTURED:
 DUMPID=003 REQUESTED BY JOB (DUMPSRV )
 DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCET CODE=MT0001   ID=1/0001

*22.25.58 DEV7          *IEA911E COMPLETE DUMP ON SYS1.DUMP05
*DUMPID=003 REQUESTED BY JOB (DUMPSRV )
*FOR ASIDS(001A,001C,001B,001E,001F,0020,001D,0022,0024,0021,0023,0028,
*0025,0029,0026)
*REMOTE DUMP FOR SYSNAME: DEV6
*INCIDENT TOKEN: DEVPLEX1 DEV6     06/10/1994 22:19:16
*ID = DUMP : APPLID IYAHZCET DUMPCODE MT0001  /1/0001
```

The dump in SYS1.DUMP05 on DEV7 was taken as a remote dump by MVS dump services as a result of the CEMT request on IYAHZCET. Note that the incident token and ID are the same as those for the dumps produced on DEV6, indicating the originating MVS and CICS IDs.

The following example lists the MVS console messages received when transaction abend SCOP is initiated after having first been added to the transaction dump table in CICS IYAHZCES as requiring related dumps. (CEMT S TRD(SCOP) ADD RELATE).

CICS IYAHZCES (ASID 1A in MVS DEV7) has at least one related task in CICS IYAHZCET (ASID 19 in MVS DEV6).

```
 23.40.41 DEV7 JOB00088  +DFHDU0201 IYAHZCES ABOUT TO TAKE SDUMP. DUMPCODE: SCOP
 23.40.49 DEV7           DFHDU0214 DFHDUMPX IS ABOUT TO REQUEST A REMOTE SDUMPX.
 23.40.55 DEV7 JOB00088  IEA794I SVC DUMP HAS CAPTURED:

 DUMPID=012 REQUESTED BY JOB (IYAHZCES)
 DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCES CODE=SCOP     ID=1/0008

 23.40.49 DEV6           DFHDU0214 DFHDUMPX IS ABOUT TO REQUEST A REMOTE SDUMPX.
 23.40.56 DEV6 JOB00029  IEA794I SVC DUMP HAS CAPTURED:
 DUMPID=007 REQUESTED BY JOB (DUMPSRV )
 DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCES CODE=SCOP     ID=1/0008


 23.41.11 DEV7 JOB00088  IEA794I SVC DUMP HAS CAPTURED:
 DUMPID=013 REQUESTED BY JOB (DUMPSRV )
 DUMP TITLE=CICS DUMP: SYSTEM=IYAHZCES CODE=SCOP     ID=1/0008

 23.41.11 DEV7 JOB00088  +DFHDU0202 IYAHZCES SDUMPX COMPLETE. SDUMPX RETURN CODE X'00'


*23.41.18 DEV6          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
*DUMPID=007 REQUESTED BY JOB (DUMPSRV )
*FOR ASID (0019)
*REMOTE DUMP FOR SYSNAME: DEV7
*INCIDENT TOKEN: DEVPLEX1 DEV7     06/10/1994 23:40:41
*ID = DUMP : APPLID IYAHZCES DUMPCODE SCOP    /1/0008
```

The dump in SYS1.DUMP03 on DEV6 was taken upon receipt of the remote dump request issued from IYAHZCES. Note the incident token and ID are the same as those for dumps produced on DEV7.

```
   *23.41.28 DEV7          *IEA911E COMPLETE DUMP ON SYS1.DUMP03
   *DUMPID=012 REQUESTED BY JOB (IYAHZCES)
   *FOR ASID (001A)
   *REMOTE DUMPS REQUESTED
   *INCIDENT TOKEN: DEVPLEX1 DEV7     06/10/1994 23:40:41
   *ID = DUMP : APPLID IYAHZCES DUMPCODE SCOP    /1/0008

   *23.41.38 DEV7          *IEA911E COMPLETE DUMP ON SYS1.DUMP04
   *DUMPID=013 REQUESTED BY JOB (DUMPSRV )
   *FOR ASID (001A)
   *REMOTE DUMP FOR SYSNAME: DEV7
   *INCIDENT TOKEN: DEVPLEX1 DEV7     06/10/1994 23:40:41
   *ID = DUMP : APPLID IYAHZCES DUMPCODE SCOP    /1/0008
```

The dump in SYS1.DUMP04 on DEV7 was taken as a remote dump by MVS dump services as a result of the request from IYAHZCES. Note the incident token and ID are the same as those for the dumps produced on DEV6, indicating the originating MVS and CICS IDs. A second dump of ASID 1A is taken because the CICS IEASDUMP does not have information indicating that a dump has already been taken for that address space.

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 that do not normally cause CICS to take a system dump.

Before you begin

To determine which messages you can do this for, look in CICS messages. 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.

About this task

You cannot enable dumping for messages that do not have the characteristics described earlier. 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.

Procedure

1. Add the dump code (constructed by removing the DFH prefix from the message number) to the system dump table.
2. Specify the SYSDUMP option.

Results

CICS then causes a system dump to be taken when the message is issued.

System dump actions with messages DFHAP0001 and DFHSR0001

In the event of a program check or MVS 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 or allowed to default. CICS determines which of these messages to issue depending on whether or not the program check or MVS 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 MVS 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 MVS 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 by a suppression of dumpcode AP0001. Note, however, that this suppresses dumps for errors in both application **and** CICS code. The XDUREQ global user exit can be used to distinguish between AP0001 situations in application and nonapplication code.

You cannot control AP0001 and SR0001 system dumps by using the DUMP parameter of the TRANSACTION resource definition. The DUMP parameter of the TRANSACTION resource definition controls only transaction dumps.

Usually, program checks, or MVS 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 a system dump is to be taken on every CICS region in the sysplex related to the CICS region on which the transaction dump is taken. A related CICS region is one on which the unit of work identifiers, in the form of APPC tokens, of one or more tasks match those in the CICS region that takes the 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 a system dump is to be taken on every CICS region in the sysplex related to the CICS region on which the system dump is taken. A related CICS region is one on which the unit of work identifiers, in the form of APPC tokens, of one or more tasks match those in the CICS region that takes the system dump.
  - 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.
  - Whether the system dump is eligible for suppression by DAE.

Note:

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 a cold start of CICS is performed, in which case they are deleted from the global catalog.

The only circumstances in which dump table additions and changes are lost are:

- When a cold start of CICS is performed.
- 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.

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.

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 increments 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 increments 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.

Note:

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.

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 used for the DAEOPTION attribute (for all new system dump codes) is set by means of the DAE= system initialization parameter. 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 1. Default values for transaction dump table entries

Action	Default	Attribute	Permitted value
Take a transaction dump?	YES	TRANDUMPING	TRANDUMP or NOTRANDUMP
Take a system dump?	NO	SYSDUMPING	SYSDUMP or NOSYSDUMP
Take system dumps on related systems?	NO	DUMPSCOPE	LOCAL or RELATED
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 2. Default values for system dump table entries

Action	Default	Attribute	Permitted value
Take a system dump?	YES	SYSDUMPING	SYSDUMP or NOSYSDUMP
Take system dumps on related systems?	NO	DUMPSCOPE	LOCAL or RELATED
Shut down CICS?	NO	SHUTOPTION	SHUTDOWN or NOSHUTDOWN
Is dump eligible for DAE?	NO	DAEOPTION	DAE or NODAE
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, and you can browse it, 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.

The transaction dump table

Table 3 on page 33 shows some examples of the sort of information that might be maintained in the transaction dump table for different transaction dump codes.

Table 3. 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
Take system dumps on related systems?	NO	YES	NO
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. System dumps on related systems are not to be taken. 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. System dumps on related systems are to be taken. 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 occur. 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). 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 main terminal operator purges a task.

  The entry shows that no action at all is to be taken in the event of such an abend. System dumps on related systems are not to be taken. 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 4 on page 34 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 4. 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		
Take system dumps on related systems?	NO		
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**, which is 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, which indicates 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 5 on page 35 shows two examples of the sort of information that might be maintained in the system dump table for different system dump codes.

Table 5. Examples of system dump table entries		
Type of information	**Example 1**	**Example 2**
System dump code	SYDMP001	MT0001
Take a system dump?	YES	YES
Take system dumps on related systems?	YES	NO
Is the dump eligible for DAE?	YES	NO
Shut down CICS?	YES	NO
Maximum times action can be taken	(default)	999
Times action already taken	0	79
System dumps taken	0	79
System dumps suppressed	0	0

The sort of information kept in the system dump table is similar to that kept in the transaction dump table (see Table 3 on page 33).

- **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. System dumps on related systems are to be taken. Dumps duplicate of this one are to be suppressed by DAE. 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 main 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.

Dumping a CFDT list structure

You can use the MVS DUMP command to obtain a dump of the coupling facility list structure for a coupling facility data table pool.

Before you begin

In order to obtain a dump of a coupling facility list structure, you must specify a value for the **DUMPSPACE** parameter in the CFRM policy for the coupling facility. The recommended value is 5% of the space in the coupling facility. For more information, see z/OS MVS Setting Up a Sysplex.

About this task

Procedure

1. Enter the following command at the console:

   ```
   DUMP COMM=(cfdt_poolname)
   ```

   In response to the DUMP command, the system prompts you with a reply number for the dump options you want to specify.
2. Enter the reply:

```
REPLY nn,STRLIST=(STRNAME=DFHCFLS_poolname,ACCESSTIME=NOLIMIT,
         (LISTNUM=ALL,ADJUNCT=DIRECTIO,ENTRYDATA=UNSERIALIZE)),END
```

If you want to use abbreviations for the keywords, enter:

```
R nn,STL=(STRNAME=DFHCFLS_poolname,ACC=NOLIM,
     (LNUM=ALL,ADJ=DIO,EDATA=UNSER)),END
```

The parameter **ACCESSTIME**=NOLIMIT allows XES to override server access time limits, to obtain serialization to take the dump. Without this parameter, no dump is taken if any server is active.

The parameters **ADJUNCT**=DIRECTIO and **ENTRYDATA**=UNSERIALIZE notify XES not to keep the serialization while dumping adjunct areas and entry data. If servers are currently active but it is considered important to obtain a serialized dump, to show the structure at a moment in time, replace these parameters with **ADJUNCT**=CAPTURE and **ENTRYDATA**=SERIALIZE. Note that this will lock out server access to the structure until the dump is complete.

If system-managed structure duplexing is available, then both instances of the structure will be dumped.

What to do next

For more information about the MVS DUMP command, see z/OS MVS System Commands.

Dumping a named counter list structure

You can use the MVS DUMP command to obtain a dump of the coupling facility list structure for a named counter pool.

Before you begin

In order to obtain a dump of a coupling facility list structure, you must specify a value for the **DUMPSPACE** parameter in the CFRM policy for the coupling facility. The recommended value is 5% of the space in the coupling facility. For more information, see z/OS MVS Setting Up a Sysplex.

About this task

Procedure

1. Enter the following command at the console:

   ```
   DUMP COMM=(named_counter_poolname)
   ```

2. In response to the DUMP command, the system prompts you with a reply number for the dump options you want to specify. Enter the reply:

   ```
   REPLY nn,STRLIST=(STRNAME=DFHNCLS_poolname,ACCESSTIME=NOLIMIT,
            (LISTNUM=ALL,ADJUNCT=DIRECTIO)),END
   ```

   Using abbreviations for the keywords, this reply can be entered as:

   ```
   R nn,STL=(STRNAME=DFHNCLS_poolname,ACC=NOLIM,(LNUM=ALL,ADJ=DIO)),END
   ```

   The parameter **ACCESSTIME**=NOLIMIT allows XES to override server access time limits, to obtain serialization to take the dump. Without this parameter, no dump is taken if any server is active.

   The parameter **ADJUNCT**=DIRECTIO notifies XES not to keep the serialization while dumping adjunct areas. If servers are currently active but it is considered important to obtain a serialized dump, to show the structure at a moment in time, replace this parameter with **ADJUNCT**=CAPTURE. Note that this will lock out server access to the structure until the dump is complete.

   If system-managed structure duplexing is available, then both instances of the structure will be dumped.

What to do next

For more information about the MVS DUMP command, see z/OS MVS System Commands.

Dumping a shared temporary storage list structure

Before you begin

In order to obtain a dump of a coupling facility list structure, you must specify a value for the **DUMPSPACE** parameter in the CFRM policy for the coupling facility. The recommended value is 5% of the space in the coupling facility. For more information, see z/OS MVS Setting Up a Sysplex.

About this task

Procedure

1. You can use the MVS DUMP command to obtain a dump of the coupling facility list structure for the shared temporary storage pool.

   You can use the MVS DUMP command to obtain a dump of the coupling facility list structure for the shared temporary storage pool. For example, enter the following command at the console:

   ```
   DUMP COMM=(sharedts_poolname)
   ```

   In response to the DUMP command, the system prompts you with a reply number for the dump options you want to specify.
2. When prompted, enter the reply:

   ```
   REPLY nn,STRLIST=(STRNAME=DFHXQLS_poolname,ACCESSTIME=NOLIMIT,
            (LISTNUM=ALL,ADJUNCT=DIRECTIO,ENTRYDATA=UNSERIALIZE)),END
   ```

   Using abbreviations for the keywords, this reply can be entered as:

   ```
   R nn,STL=(STRNAME=DFHXQLS_poolname,ACC=NOLIM,
        (LNUM=ALL,ADJ=DIO,EDATA=UNSER)),END
   ```

   The parameter **ACCESSTIME**=NOLIMIT allows XES to override server access time limits, to obtain serialization to take the dump. Without this parameter, no dump is taken if any server is active.

   The parameters **ADJUNCT**=DIRECTIO and **ENTRYDATA**=UNSERIALIZE notify XES not to keep the serialization while dumping adjunct areas and entry data. If servers are currently active but it is considered important to obtain a serialized dump, to show the structure at a moment in time, replace these parameters with **ADJUNCT**=CAPTURE and **ENTRYDATA**=SERIALIZE. Note that this will lock out server access to the structure until the dump is complete.

   If system-managed structure duplexing is available, then both instances of the structure will be dumped.

What to do next

For more information about the MVS DUMP command, see z/OS MVS System Commands.

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 6 on page 38 shows

the circumstances in which dumps are invoked, the modules that invoke them, and the corresponding dump codes.

Table 6. ZCQTRACE dump codes

Module	Dump code	When invoked
DFHTRZCP	AZCQ	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.

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.

Formatting and interpreting dumps

CICS system dumps and transaction dumps are written unformatted 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” on page 38 and “Formatting system dumps” on page 39, respectively.

Formatting transaction dumps

You can format transaction dumps offline using the CICS dump utility program, DFHDU710. Individual transaction dumps must be formatted in their entirety, but you can control which dumps are formatted from the dump data set.

About this task

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 ID - 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 DFHDU710 to format transaction dumps, see the Dump utilities (DFHDU680 and DFHPD680).

Formatting system dumps

You can process system dumps from the dump data set by invoking the interactive problem control system (IPCS).

About this task

The CICS formatting routine for use under the MVS interactive problem control system (IPCS) is supplied as DFHPDX. This standard name is not suitable for those users running more than one release of CICS, because the dump formatting process in each version of DFHPDX is release-specific, and you must use the correct version for the system dump you are formatting. The module is named with the release identifier as part of the name - DFHPD710 is the formatting routine you must define to IPCS when formatting CICS TS 5.4 system dumps.

The DFHIPCSP CICS exit control data

IPCS provides an exit control table with imbed statements to enable other products to supply exit control information.

The IPCS default table, BLSCECT, normally in SYS1.PARMLIB, has the following entry for CICS:

```
IMBED MEMBER(DFHIPCSP)  ENVIRONMENT(ALL) /*CICS                 */
```

Ensure that the CICS-supplied DFHIPCSP member can be found by your IPCS job. You can either copy DFHIPCSP into SYS1.PARMLIB (so that it is in the same default library as BLSCECT) or provide an IPCSPARM DD statement to specify the library containing the IPCS control tables. For example:

```
//IPCSPARM  DD  DSN=SYS1.PARMLIB,DISP=SHR         For BLSCECT
//          DD  DSN=CICSTS54.CICS.SDFHPARM,DISP=SHR     For DFHIPCSP
```

Figure 1 on page 39 shows the release-specific entries specified in DFHIPCSP.

```
 /* ============================================================== */
     EXIT EP(DFHPD660) VERB(CICS660) ABSTRACT(+
    'CICS Transaction Server for z/OS V4 R1 analysis')
   EXIT EP(DFHPD670) VERB(CICS670) ABSTRACT(+
    'CICS Transaction Server for z/OS V4 R2 analysis')
   EXIT EP(DFHPD680) VERB(CICS680) ABSTRACT(+
    'CICS Transaction Server for z/OS V5 R1 analysis')
   EXIT EP(DFHPD690) VERB(CICS690) ABSTRACT(+
    'CICS Transaction Server for z/OS V5 R2 analysis')
   EXIT EP(DFHPD700) VERB(CICS700) ABSTRACT(+
    'CICS Transaction Server for z/OS V5 R3 analysis')
   EXIT EP(DFHPD710) VERB(CICS710) ABSTRACT(+
    'CICS Transaction Server for z/OS V5 R4 analysis')
/* ============================================================== */
```

Figure 1. Release-specific entries in DFHIPCSP for DFHPDnnn routines

To use the DFHIPCSP member as it is, rename the CICS-supplied version of DFHPDX for earlier releases to the names shown in the table.

- The IPCS dump analysis subcommands enable you to format and analyze CICS-produced SDUMPs, which you can either view at a terminal or print. You can:
  - Examine the data in a dump
  - Locate and verify control blocks associated with certain functions or system components
  - Trace and verify chains of control blocks
  - Perform contention analysis on key MVS resources
  - Locate modules and unit control blocks
  - Execute user-written exits for certain control blocks

  - Keep a list of names and locations of control blocks and areas of the dump that you consider important.

- The CICS*nnn* dump exit (where *nnn* is the CICS release identifier) enables you to format a dump selectively by specifying one or more CICS component identifiers as parameters to the exit. Thus, the CICS710 dump exit enables you to process a CICS TS for z/OS, Version 5.4 dump selectively. You can:
  - Specify which job in the dump data set is to be formatted (**JOB** parameter).
  - Specify which component storage areas are to be formatted, and at what level of detail, by using formatting keywords and level numbers (**keyword** parameter). You do this using the IPCS command: `VERBEXIT CICS710 'keyword,...'`.
  - 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).

The use of formatting keywords enables you to format those parts of the dump that interest you at any particular time, at specific levels of detail. You have the option of formatting other parts later for further investigation by you or by the IBM service organizations. It is advisable to copy your dumps so that you can save the copy and free the dump data set for subsequent use.

Summary of system dump formatting keywords and levels

The component keywords specify the areas of CICS for which you want the CICS710 exit to format dump data, and the level number operand specifies the amount of data that 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 that 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.

APS=<TASKID=Task identifier>
Application selection. The APS component keyword allows you to limit the formatting of system dumps to only those storage areas relating to the task identifier specified. Contents of the application domain control blocks for the specified transaction will be listed along with language environment storage areas for the same transaction. You must use angled brackets around the specified parameter.

AS [={0|1|2|3}]
Asynchronous services domain.

AS=0
Suppress formatting of asynchronous services domain control blocks.

AS=1
Format a summary of addresses of the AS control blocks for each active transaction.

AS=2
Format the contents of the AS control blocks for each active transaction.

AS=3
Format level-1 and level-2 data.

BA [={0|1|2|3}]
Business application manager domain.

BA=0
Suppress formatting of business application manager domain control blocks.

BA=1
Format the BA unit of work context (BAUW) and BA activity (BAACT) summaries.

BA=2
Format the anchor block (BADMANC), BA unit of work context (BAUW), and BA activities (BAACT).

BA=3
Format level-1 and level-2 data.

BR [={0|1|2|3}]
The 3270 bridge domain

BR=0
Suppress formatting of bridge domain control blocks.

BR=1
Format bridge facility summary information.

BR=2
Format all control blocks and bridge messages in the system.

BR=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.

CQ [={0|1|2}]
Console queue.

CQ=0
Suppress formatting of console queue control blocks.

CQ=1
Format the console queue summary.

CQ=2
Format the console queue control blocks and CQ trace table.

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).

CV[={0|1|2|3}]
The CCSID conversion interface.

CV=0
Suppress formatting of CCSID conversion control blocks.

CV=1
Format the summary of CCSID conversion information.

CV=2
Format the CCSID conversion control blocks.

CV=3
Format level-1 and level-2 data.

DB2 [={0|1|2|3}]
The CICS DB2 interface.

DB2=0
Suppress formatting of DB2 control blocks.

DB2=1
Format the summary of tasks currently using the CICS DB2 interface.

DB2=2
Format the control blocks.

DB2=3
Format level-1 and level-2 data.

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.

DH[={0|1|2|3}]
The document handler domain.

DH=0
Suppress formatting of DH control blocks.

DH=1
Format document handler summary information.

DH=2
Format the domain anchor block, document anchor block, document control record, document data block, and document bookmark block.

DH=3
Format level-1 and level-2 data.

DLI[={0|2}]
CICS DL/I interface.

DLI=0
Suppress formatting of DLI control blocks.

DLI=2
Format DLI control blocks.

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.

DP[={0|1|2|3}]
The debugging profiles domain.

DP=0
Suppress formatting of DP control blocks.

DP=1
Format a summary of the DP control blocks.

DP=2
Format the DP control blocks.

DP=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|1|2|3}]
The dump domain.

DU=0
Suppress formatting of DU control blocks.

DU=1

Format the dump domain summary information.

DU=2
Format the DU anchor block.

DU=3
Format level-1 and level-2 data.

EC [={0|1|2|3}]
The event capture domain.

EC=0
Suppress formatting of EP control blocks.

EC=1
Format summary information.

EC=2
Format EP control blocks.

EC=3
Format level-1 and level-2 data.

EJ[={0|1}]
The enterprise Java domain.

EJ=0
Suppress formatting of EJ control blocks.

EJ=1
Format the EJ control blocks.

EM[={0|1|2}]
The event manager.

EM=0
Suppress formatting of EM control blocks.

EM=1
Format a summary of the active event pools.

EM=2
Format the content of the EM control blocks.

EP [={0|1|2|3}]
The event processing domain.

EP=0
Suppress formatting of EP control blocks.

EP=1
Format summary information.

EP=2
Format EP control blocks.

EP=3
Format level-1 and level-2 data.

FCP[={0|2}]
The file control program.

FCP=0
Suppress formatting of the file control table.

FCP=2
Format the file control table.

FT[={0|1|2|3}]
The feature domain.

FT=0
Suppress formatting of the feature table.

FT=1
Provide a system dump summary.

FT=2
Provide a dump for the feature table.

FT=3
Provide a summary and dump for the feature table.

ICP[={0|2}]
The interval control program.

ICP=0
Suppress formatting of the interval control elements (ICEs).

ICP=2
Format the ICEs.

IE[={0|1|2}]
The ECI over TCP/IP domain.

IE=0
Suppress formatting of the IE control blocks.

IE=1
Format a summary of the IE control blocks.

IE=2
Format the content of the IE control blocks.

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.

IPCS does not produce page numbers if formatting directly to the terminal.

IS[={0|1|2|3}]
The IP interconnectivity domain.

IS=0
Suppress formatting of IS domain information.

IS=1
Format the summary of IPCONN definitions and their sessions.

IS=2
Format the IS domain control blocks.

IS=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.

LG[={0|1|2|3}]
The log manager domain.

LG=0
Suppress formatting of log manager domain control blocks.

LG=1
Format the log manager summary.

LG=2
Format all log manager control blocks.

LG=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.

ML[={0|1|2|3}]
The markup language domain.

ML=0
Suppress formatting of the ML domain control blocks.

ML=1
Format summary information.

ML=2
Format ML domain control blocks.

ML=3
Format level-1 and level-2 data.

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.

MP[={0|1|2|3}]
The managed platform domain.

MP=0
Suppress formatting of managed platform domain control blocks.

MP=1
Format the summary information for MP domain.

MP=2
Format the MP domain control blocks.

MP=3
Format level-1 and level-2 data.

MQ[={0|1|2|3}]
The CICS-IBM MQ interface.

MQ=0
Suppress formatting of CICS-IBM MQ control blocks.

MQ=1
Format the summary of tasks currently using the CICS-IBM MQ interface.

MQ=2
Format the control blocks.

MQ=3
Format level-1 and level-2 data.

MRO[={0|2}]
CICS multiregion operation.

MRO=0
Suppress formatting of all MRO control blocks.

MRO=1
Format MRO control block summary.

MRO=2
Format MRO control blocks, APPC URDs, and any associated DWE chains.

MRO=3
Format level-1 and level-2 data.

NQ [={0|2}]
The enqueue manager domain.

NQ=0
Suppress formatting of NQ control blocks.

NQ=2
Format NQ control blocks.

OT[={0|1|2}]
The object transaction domain.

OT=0
Suppress formatting of OT control blocks.

OT=1
Format a summary of OT control blocks.

OT=2
Format the contents of the OT control blocks.

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 (for TRANSACTION resource definitions).

PCT=0
Suppress formatting of the transaction resource definitions.

PCT=2
Format the transaction resource 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 PROGRAM resource definitions, the PLCBs, and the HTBs.

PG=3
Format level-1 and level-2 data.

PI [={0|1|2|3}]
The pipeline domain.

PI=0
Suppress formatting of PI domain information.

PI=1
Format the pipeline summary.

PI=2
Format the PI domain control blocks.

PI=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.

PT [={0|1|2|3}]
The partner domain.

PT=0
Suppress formatting of partner domain control blocks.

PT=1
Format summary information.

PT=2
Format all control blocks.

PT=3
Format level-1 and level-2 data.

RD [={0|2}]
Resource definition.

RD=0
Suppress formatting of RD areas.

RD=2
Format the RD recovery and locking blocks.

RL [={0|1|2|3}]
The resource life-cycle domain.

RL=0
Suppress formatting of RL control blocks.

RL=1
Format summary information.

RL=2
Format RL control blocks.

RL=3
Format level-1 and level-2 data.

RM [={0|2}]
The recovery manager domain.

RM=0
Suppress formatting of RM control blocks.

RM=2
Format RM control blocks.

RS [={0|1|2|3}]
The region status domain.

RS=0
Suppress formatting of RS control blocks.

RS=1
Format summary information.

RS=2
Format RS control blocks.

RS=3
Format level-1 and level-2 data.

RX [={0|2|3}]
The recoverable resource manaagement services (RRMS) domain.

RX=0
Suppress formatting of RRMS control blocks.

RX=1
Format summary of unit-of-recovery information.

RX=2
Format all RRMS domain storage.

RX=3
Format level-1 and level-2 data.

RZ[={0|1|2}]
The request streams domain.

RZ=0
Suppress formatting of EJ control blocks.

RZ=1
Format a summary of RZ control blocks.

RZ=2
Format the contents of the RZ control blocks.

RZ=3
Format level-1 and level-2 data.

SH [={0|1|2|3}]
The scheduler services manager domain.

SH=0
Suppress formatting of the scheduler services manager domain control blocks.

SH=1
Format the SH unit of work pending queue (SHPQUEUE), and the bound, pending, and committed SH request (SHREQUES) summaries.

SH=2
Format the anchor block (SHDMANC), SH unit of work pending queue, (SHPQUEUE) and the bound, pending, and committed SH requests (SHREQUES).

SH=3
Format level-1 and level-2 data.

SJ[={0|1|2|3}]
The JVM domain.

SJ=0
Suppress formatting of SJ control blocks.

SJ=1
Format a summary of SJ control blocks.

SJ=2
Format the contents of the SJ control blocks.

SJ=3
Format level-1 and level-2 data.

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.

SO[={0|1|2|3}]
The socket domain.

SO=0
Suppress formatting of the socket domain control blocks.

SO=1
Format a summary of the socket domain control blocks.

SO=2
Format the contents of the socket domain control blocks in full.

SO=3
Format both the level-1 and level-2 data. Specifying SO is the same as SO=3.

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.

TCP[={0|1|2|3}]
The terminal control program.

TCP=0
Suppress formatting of TCP control blocks.

TCP=1
Format the terminal control and AID summaries.

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 queue definitions.

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.

TK[={0|1|2|3}]
The task summary and dump formatter.

TK=0
Suppress task summary formatter.

TK=1
Format a summary table for all tasks in the system dump.

TK=2
Format the transaction (TXN), user and system task control areas (TCAs), exec interface block (EIB), exec interface user structure (EIUS), kernel task control block (KTCB), transaction monitoring area (TMA) and program level control blocks (PLCBs) for each of the tasks in the system dump.

TK=3
Format level-1 and level-2 data.

TKS=<TASKID=Task Identifier>
Format the level-1 and level-2 data for the specified task.

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.

You can use the TRS component keyword to select trace entries from internal trace in a system dump for formatting and printing. You do this in a similar way to the selection of trace entries in an auxiliary trace for formatting and printing.

The trace selectivity parameter can be any valid trace selectivity parameter available to DFHTU710 for the formatting of CICS auxiliary trace entries, except for the parameters PAGESIZE, ABBREV, SHORT

and FULL. You can also use the LAST_BLOCKS parameter, which is specifically for formatting internal trace in a system dump. As with DFHTU710, you can select any number of parameters from those available.

The trace selection parameters have the same format and default values when used to select trace entries from an internal SDUMP trace, as when you use DFHTU710 to format auxiliary trace entries. You must use angled brackets around the parameter, or sequence of parameters, that you specify.

TS[={0|1|2|3}]
Temporary storage domain.

TS=0
Suppress formatting of temporary storage control blocks.

TS=1
Format a summary of temporary storage control blocks and temporary storage control block checking.

TS=2
Format temporary storage control blocks.

TS=3
Format level-1 and level-2 data.

You can also specify the options for TS with angled brackets, for:

TS=<1>
Summary.

TS=<2>
Format control blocks.

TS=<3>
Consistency checking of the TS buffers with the TS control blocks.

You can specify more than one of these values between angled brackets. For example, TS=<1,2> gives summary and formatting of control blocks without consistency checking.

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.

WB[={0|1|2}]
The Web domain.

WB=0
Suppress formatting of Web domain control blocks.

WB=1
Format the Web domain summary. This level displays the current state of CICS Web support, followed by a summary of the state blocks controlled by the state manager.

WB=2
Format the control blocks. This level displays the current state of CICS Web support, followed by the Web anchor block, the global work area and associated control blocks, and the Web state manager control blocks.

WU[={0|1|2|3}]
The CICS management client interface.

WU=0
Suppress formatting of CICS management client interface control blocks.

WU=1
Format the CICS management client interface summary. This level displays a list of currently running CICS management client interface requests and a list of all cached results currently being held.

WU=2
Format the control blocks. This level displays the CICS management client interface anchor block and a control block for each running request.

WU=3
Format level-1 and level-2 data.

W2[={0|1|2|3}]
The Web 2.0 domain.

W2=0
Suppress formatting of Web 2.0 domain control blocks.

W2=1
Format the Web 2.0 domain summary.

W2=2
Format the control blocks.

W2=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|2}]
The security domain.

XS=0
Suppress formatting of anchor block and supervisor storage.

XS=2
Format anchor block and supervisor storage.

For a more detailed list of the contents of SDUMPs for each of the VERBEXIT keywords, see SDUMP contents and IPCS CICS VERBEXIT keywords.

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.

Interpreting transaction dumps

The contents of a transaction dump are described, and guidance on locating information that is useful for debugging transactions is provided.

About this task

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 users should be using 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 are not described in this section.

The job log for the dump utility program, DFHDU710, 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.

Procedure

1. The first thing to look for is the 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.
2. Look for the CICS Transaction Server for z/OS level.

   It shows you what level of CICS was being executed when the transaction dump was taken.
3. The transaction environment summary provides you with a formatted summary of the transaction environment at the time the dump is taken.

4. Depending on whether the transaction abended remotely or locally, you will see different information in the transaction dump as follows:
   a) If the transaction abended remotely (the abend originally occurred in a remote distributed program link (DPL) server program), and the abend is being reissued 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.
   b) If the transaction dump was taken in response to a local abend with abend code AICA, ASRA, ASRB, or ASRD:
      i) 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.
      ii) A set of registers that belong to the program that was executing when the error was detected is also provided. They are taken from the transaction abend control block.
   c) If the transaction abended locally with abend code ASRA or ASRB:
      i) The execution key that is in force at the time of the abend is formatted. It is taken from the transaction abend control block.
      ii) CICS formats the space, basespace or subspace, in which the program was executing. If transaction isolation is not enabled, the program always executes in the basespace.

   See "Transaction storage" on page 57 for more information on the transaction abend control block.
5. 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.
6. 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.

   The system area of the task control area is formatted separately, because 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.
7. Any transaction work area relating to the transaction is formatted, if present.
8. The EXEC interface structure (EIS) contains information about the transaction and program specific to the execution interface component of CICS.
   a) The system EXEC interface block (SYSEIB) is used solely by programs using the SYSEIB option. If you see this in the transaction dump, read Defining translator options
   b) The EXEC interface user structure (EIUS) contains execution interface component information that must reside in user key storage.
9. 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 Function codes of EXEC CICS commands.
10. 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.
11. The common system area (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.

The common system area optional features list (CSAOPFL), an extension of the CSA, contains the addresses of CICS optional features.

12. The abbreviated-format trace table is formatted by default. You can suppress it by specifying the **NOABBREV** parameter in the DFHDU710 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 58.
13. 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 DFHDU710 job. It contains more detail, but you can probably get all the information you need using the abbreviated trace.
14. The common work area (CWA) is the installation-defined work area for use by all programs and is formatted if it exists.
15. You are likely to find several areas of the dump that describe transaction storage that can be useful for debugging purposes.

    For information on what to look for, see “Transaction storage” on page 57.
16. The TCTTE (terminal control table terminal entry) 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 might find more than one.

    To look at the TIOA for the task, find the address in field TCTTEDA of the TCTTE.
17. Program information for the current transaction 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 58.

    Other program manager control blocks are shown too, including resource definition information for active programs, and load list elements and program storage for any programs loaded by this transaction but not yet released.

    For each program level you can find the current channel (if any), other channels created by the link level, and all their containers. Up to 32K of each container's data is also displayed. Note that this is a copy of the data rather than the actual address of the data.

    For each task you can find the transaction channel (if any), and all its containers. Up to 32K of each container's data is also displayed. Note that this is a copy of the data rather than the actual address of the data.
18. 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.

Transaction storage

Transaction storage is storage that is either obtained by CICS to store information about a transaction, or obtained explicitly by the transaction, by using a GETMAIN request, for its own purposes.

The dump might contain several such areas, each introduced by a header that describes it as transaction storage of a particular class, for example:

USER24
USER31
USER64
CICS24
CICS31
CICS64

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 the following useful valuable information that relates to the abend:

- The program status word (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.) Registers 12 and 13 contain the addresses of the TCA and CSA, respectively, in all abends except for ASRA and ASRB abends. For ASRA and ASRB abends, these registers contain data as at the time of the abend.
- The name of the failing program.
- The offset in 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).
- Whether the program is executing in a subspace or the basespace.
- The subspace STOKEN.
- The subspace's associated ALET.

If the abend originally occurred in a remote DPL server program, an eye-catcher ***REMOTE*** is present, and the registers and PSW are not present.

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.

Before you begin

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" on page 58.

About this task

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 59.

Last command identification

This process for identifying the last command applies to system dumps, in which it is necessary to identify the abending task.

Step 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 Function codes of EXEC CICS commands. The AP 00E1 trace point also includes the value of register 14 in the RET= field. Register 14 is the return address of the calling routine. See “Formatting and interpreting trace entries” on page 78.

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. You next need to 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 CICS transaction storage areas of the transaction dump. These areas are maintained by CICS, and they relate to the transaction that was running when the abend occurred. Find the eye-catcher **DFHTACB** in at least one of the areas. This eye-catcher signifies the start of the transaction abend control block and it contains the registers and PSW belonging to the program that was running 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 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 shows a return address into your program. Either way, correlate the address with a statement in the source code for the program.

   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, refer to a compiler output mapping source statements onto the object code.

Locating program data

You might want to look at the data that your application program has in its storage areas.

You must refer to the appropriate programming language information for details of the structure of the acquired storage of the program.

CICS maintains a pointer to the chain of dynamic storage that an assembler program uses in field TCAPCDSA of the system area of the TCA.

Dumps for C/370 programs

The use of the relevant C/370 registers is as follows:

Register	Use
3	In most circumstances, this is the base register
12	Holds the address of the CICS TCA for the C/370 program
13	Holds the address of the register save area

Location of COBOL dumped areas

The dumped COBOL program is in the "program information for the current transaction" section of the dump, and 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 contains the following:

- The COBOL working storage for the task
- A copy of the TGT

The TGT is 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 current registers of the program are stored on any request for CICS service.

Storage freeze

Certain 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 CSFE - terminal and system test.

Formatting a coupling facility data table pool dump

About this task

Procedure

1. To format a coupling facility data table structure dump, use the IPCS STRDATA subcommand.
2. To display the table index list, use the STRDATA subcommand as follows:

   ```
   STRDATA DETAIL LISTNUM(2) ENTRYPOS(ALL)
   ```

   The key of each entry in this list is the table name, and the first word of the adjunct area is the corresponding data list number. If the table is open, entry data is present containing a list of information about the current table users (regions that have the table open). Each one is identified by its MVS system name and CICS APPLID. The number of users is at +X'14' in the adjunct area. After any valid table user elements, the rest of the data area is uninitialized and can contain residual data up to the next 256-byte boundary.
3. To display the table data, convert the data list number to decimal and specifying it on another STRDATA subcommand.

For example, if the first word of the adjunct area is X'00000027', the command to display the table data is as follows:

```
STRDATA DETAIL LISTNUM(39) ENTRYPOS(ALL)
```

In the data list, the key of each entry is the record key, and the data portion contains the user data with a 2-byte length prefix (or a 1-byte X'80' prefix if the data length is 32767 bytes). The rest of any data area is uninitialized and can contain residual data up to the next 256-byte boundary. The entry version contains the time stamp at the time the record was last modified or created.

The adjunct area contains information for locking and recovery. It contains null values (binary zeros) if the record is not locked. When the record is locked, the lock owner APPLID and UOWID appear in this area.

If a record has a recoverable rewrite pending, there are two entries with the same key, where the second entry is the before-image.

What to do next

For information about the STRDATA subcommand and its options, see z/OS MVS IPCS User's Guide.

Formatting a named counter pool dump

About this task

Procedure

To format a named counter structure dump, use the IPCS STRDATA subcommand as follows:

```
STRDATA DETAIL LISTNUM(0) ENTRYPOS(ALL)
```

Results

The key of each entry in this list is the counter name. The version field contains the counter value minus its maximum value minus 3 (in twos complement form) which has the effect that all counters have a value of -2 when they have reached their limit (and the value of -1, equal to all high values, never occurs). The start of the adjunct area contains the 8-byte minimum and maximum limit values that apply to the counter.

What to do next

For information about the STRDATA subcommand and its options, see z/OS MVS IPCS User's Guide.

Formatting a shared temporary storage pool dump

Procedure

1. To format a shared temporary storage pool dump, use the IPCS STRDATA subcommand.
2. To display the queue index list, use the STRDATA subcommand as follows:

   ```
   STRDATA DETAIL LISTNUM(2) ENTRYPOS(ALL)
   ```

   The key of each entry in this list is the queue name. For a small queue, the first word of the adjunct area is the total length of queue data which is included in the queue index entry; for a queue that has been converted to the large format, the first word contains zero. The second word of the adjunct area is the number of the corresponding data list if the queue has been converted to the large format, or zero otherwise.

   For a small queue, the queue data is stored as the data portion of the queue index entry, with a 2-byte length prefix preceding each item.

3. To display the queue data for a large queue, convert the data list number to decimal and specifying it on another STRDATA subcommand
   For example if the second word of the adjunct area is X'0000000A', the command to display the queue data is:

   ```
   STRDATA DETAIL LISTNUM(10) ENTRYPOS(ALL)
   ```

   In the data list, the key of each entry is the item number, and the data portion contains the item data with a 2-byte length prefix. The rest of any data area is uninitialized and may contain residual data up to the next 256-byte boundary.

What to do next

For information about the STRDATA subcommand and its options, see z/OS MVS IPCS User's Guide.

Statistics

Statistics are often overlooked as a source of debugging information, but statistics that relate to an application program can help solve problems.

It is useful to have a statistical profile (as mentioned in “Documentation” on page 17) to use for problem determination. If you compare the information in the profile with the statistical information produced by CICS, any differences you find might 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 might notice from these statistics that tables are being loaded, or programs are being linked, for which there is no known requirement.

You can also use statistics to check terminals, files, queues, and other resources, for irregularities in their activity. For example, if a terminal has a number of errors that are recorded for a particular transaction that equal the number of times that transaction was run, this might indicate that an incorrect data stream is being sent to that terminal. For more information about statistics, see Introduction to CICS statistics in Monitoring.

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 in CICS code itself, you can define user event monitoring points in your own application programs by using the **MONITOR** POINT 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 could 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 about the MCT, see Monitoring control table (MCT). For programming information about the **MONITOR** POINT command, see MONITOR.

For guidance about choosing performance tools, see Performance measurement tools. For information about the transactions needed to invoke these tools, see CEMT - main terminal.

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 important in solving problems, because it can help you determine what data was entered just before the transaction failed, and if there is any output.

The more you know about the information that was input at the terminal on which the transaction failed, the better your chance of duplicating the problem in a test environment. However, this information might 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 output from a transaction is sometimes easier to capture. If you have a locally attached printer, you can make a copy. (The problem might be that the printer output is incorrect.)

The items to look for on the input side are:

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

On the output screen, check the following points:

1. Do all the required fields contain data?
2. Is the data correct?
3. Is the screen format as it was designed?
4. Are there any non-display fields used to pass data that might 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 what you expect. You can use the CICS-supplied transaction, CEBR, to inspect temporary storage queues in some detail.

See CEBR - temporary storage browse in the IBM Knowledge Center 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 (and CDBC if you use DBCTL) to see if there are any relevant messages.

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

1. Are the required entries there?
2. Are the entries in the correct order?
3. Is the queue being written the same one that is being read?

Passed information

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 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.

Investigating interval control waits

If you have a task that is not running and interval control seems to be involved, you can use this information to understand the possible causes.

About this task

The following is a list of possible causes, and suggestions to consider before you carry out a detailed investigation. If these do not give you enough information in order to solve the problem, go to "Finding the reason for a DELAY request not completing" on page 135 for further guidance. If, in the course of your preliminary investigations, you find that the task is waiting because the terminal where it is due to start is unavailable, turn to "Investigating terminal waits" on page 118.

Procedure

- A terminal task that should have been initiated with an **EXEC CICS START** command did not start when you expected it to. **CEMT INQ TASK** does not recognize the task, because it has not yet been attached.

  One approach is to identify the terminal where the subject task should have started, and see if that terminal is unavailable. You can use **CEMT INQ TERMINAL** to find the status of the terminal.
- You have found that 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.

  a) Find the target TERMID for other tasks issuing **EXEC CICS START** commands to supply more data.

  The resource name gives you the name of the terminal running the task in the ICGTWAIT and therefore the target TERMID.

  b) If there are no tasks in the system that would issue START commands for this TERMID, you need to determine whether this is reasonable.

  c) 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.

  d) Look at the deadlock timeout 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, 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.

  a) 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, that suggests a possible CICS error.

b) Consider 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:

  a) Check to make sure that the interval or time specified on the POST was what you intended.

  b) If it is, check to see whether the ECB being waited on was posted. If it has been posted, that indicates a possible CICS error.

Results

If none of the simple checks outlined here help you to solve the problem, read the following information.

Using CICS trace

General CICS tracing is handled by the CICS trace domain. It traces the flow of execution through CICS code, and through your applications as well. You can see what functions are being performed, which parameters are being passed, and the values of important data fields at the time trace calls are made. This type of tracing is also useful in first failure data capture, if an exception condition is detected by CICS.

For programming information about how to make trace calls from your own application programs, see CICS command summary.

Trace points are included at specific points in CICS code. From these points, trace entries can be written to any currently selected trace destination. 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.

CICS provides different levels of tracing to assist with problem determination. Standard trace level 1 is the default setting for each component to be traced within CICS. The user can use CETR to specify what trace levels are set for each component of CICS. By default, INTTR, SYSTR, and USERTR are set ON. This means the main system and user trace flags default to be set on, and internal tracing is active. STNTR defaults to 1, as do all the STNTRxx values, and as a result standard trace component tracing defaults to level 1 for all CICS trace components. The consequence of this is that by default a CICS system incurs the CPU usage to provide this level of internal CICS trace data.

There is a trade-off between the CPU cost in capturing trace data for problem determination, set against the ability to diagnose problems if they occur. Some customers elect to run with limited levels of tracing active on their system. While choosing to use CICS trace does increase processing requirements, not using CICS tracing reduces the amount of problem determination information that is available for the CICS region.

Note: CICS always performs exception tracing when it detects an exception condition, so a minimum of first failure data capture is provided regardless of your CICS trace setting. However, exception tracing by its nature is limited in what diagnostic data it can provide. It is difficult to perform initial problem determination without CICS tracing being active and all components capturing their trace data, as it is the trace information that helps to identify the flow of system activity, and the events in chronological order, leading up to a failure. For this reason, to assist with problem determination it is recommended that the default settings of all CICS domains and components is used when tracing is active. This is standard trace level 1 tracing.

All CICS trace points are listed in alphabetical sequence in Trace entries.

Trace levels

Apart from exception trace points, all trace points in CICS have an associated level attribute. The level of a trace point indicates the depth of information that the trace point provides.

Trace levels 1–32 are available in CICS, but in practice nearly all mainline trace points have a trace level of 1 or 2.

You can use the trace levels to specify the level of CICS system tracing that you require for the CICS region, or for an individual component or task.

Level-1 trace points
Level-1 trace points are designed to give you enough diagnostic information to fix errors caused by user applications or user actions. CICS provides level-1 trace points in the following situations:

- Entry to, and exit from, every CICS domain. The information includes the domain call parameter list, and the address of any data that is useful for a high-level understanding of the function to be performed.
- 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, such as the z/OS Communications Server. The information includes the request that is to be made, the input parameters on the request, and the result of the call.

Level-2 trace points
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
Trace points at level 3 and above are reserved for special cases. Very few components have trace points higher than level 2, and they are only likely to be of use by IBM support staff.

Trace destinations

You can choose from any number of the destinations for the trace entries produced by CICS. Any combination of these destinations can be active at any time.

- The internal trace table
- The auxiliary trace data sets
- The MVS generalized trace facility (GTF) data sets
- The JVM server trace file in z/OS Unix System Services

You can choose the most appropriate trace destinations based on their characteristics, the amount of trace data that you need to capture, and whether you want to integrate CICS tracing with tracing done by other programs. The JVM server trace file is unique to each JVM server. For more information on this trace file, see Troubleshooting Java applications.

You can control the status and certain other attributes of the trace destinations either while CICS is running, or by specifying system initialization parameters at CICS startup. For instructions to control the trace destinations, see “Setting trace destinations and tracing status” on page 77.

Internal trace table

The CICS internal trace table is held in storage in the CICS region. It is allocated at an early stage during CICS initialization, and it exists for the whole of the CICS run.

You use the **TRTABSZ** system initialization parameter to specify the size of the internal trace table at CICS startup. Its minimum size is 16 KB, and its maximum size is 1 GB. The default size is 12288 KB (12 MB).

Every CICS region always has an internal trace table. Even if internal tracing has not been started for the CICS region, the internal trace table is used as a buffer for the other trace destinations. Trace entries are built there and copied to the auxiliary trace data sets or to GTF trace if those destinations are started.

In addition, exception trace entries are always written to the internal trace table, even if no trace destinations are currently started. Other trace destinations that are currently started 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 function provides first failure data capture.

The internal trace table wraps when it is full. When the end of the table is reached, the next trace entry to be directed to the internal trace table goes to the start, and overlays the trace entry that was formerly there. You can increase or decrease the size of the internal trace table 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.

For a transaction dump, CICS copies the current internal trace table to produce the transaction dump trace table. You use the **TRTRANSZ** system initialization parameter to specify the size of the transaction dump trace table, which is created in MVS storage in 64-bit (above-the-bar) storage.

The internal trace table 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 trace entries over a short period, one of the other trace destinations is likely to be more appropriate.

You can format the internal trace table in two ways:

- From a CICS system dump, using the CICS print dump exit, DFHPD710.

  If the internal trace table is large, you can use trace selection parameters to reduce the number of trace entries that are formatted. See Selecting parts of the CICS internal trace table.
- From a transaction dump, using the CICS dump utility program, DFHDU710.

Auxiliary trace data sets

The auxiliary trace data sets are CICS-owned BSAM data sets named DFHAUXT and DFHBUXT. If you want to use auxiliary trace, you must create one or both of these data sets before you start CICS; you cannot create them while CICS is running.

For instructions to set up the auxiliary trace data sets, see Setting up auxiliary trace data sets. The auxiliary trace data sets require two 4 KB data buffers in the CICS region's storage.

When you first start 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.

When you have two auxiliary trace data sets, you can choose from the following actions for CICS to take when one data set is full. You specify your chosen action using the auxiliary switch, which you can set using the AUXTRSW system initialization parameter or the CEMT transaction:

- When the initial data set is full, no more trace entries are directed to the auxiliary trace data sets. (AUXTRSW=NO)
- When the initial data set is full, then the other data set receives the next trace entries. When that one is full, no more trace entries are directed to the auxiliary trace data sets. (AUXTRSW=NEXT)
- Auxiliary trace data is written alternately to each data set, and CICS switches from one to the other every time the current one becomes full. With this action selected, trace entries are eventually overwritten, as they are in the internal trace table. (AUXTRSW=ALL)

When auxiliary tracing is started or when it is paused, the auxiliary trace data set that is currently in use is open. When auxiliary tracing is stopped, the auxiliary trace data set is closed.

You can use auxiliary trace data sets to collect large amounts of trace data, provided that you initially define large enough data sets. For example, you might want to use auxiliary trace to track system activity over a long period of time, perhaps to solve an unpredictable storage violation problem. Auxiliary trace

can be particularly useful if you are using CICS trace during startup, because of the high volume of trace entries that are written when CICS is initializing.

CICS provides a trace utility program, DFHTU710, that you can use to extract all or selected trace entries from auxiliary trace data sets, and format and print the data. You can select the trace entries based on the time when they were written, and the reports from the trace utility program include time stamps to help you match external events with a particular area of the trace.

Generalized Trace Facility (GTF)

The generalized trace facility (GTF) in z/OS is a service aid that is part of the MVS system product. CICS issues an MVS GTRACE macro to write trace entries to GTF. GTF must be started with the TRACE=USR option in MVS before you start CICS GTF trace.

GTF stores trace entries in a trace table in MVS main storage, or in up to 16 data sets on tape or disk.

If the storage available to GTF becomes full, the next trace entries go to the start, and overlay the trace entries that were formerly there. For more information about setting up GTF in z/OS, see The Generalized Trace Facility (GTF) in z/OS MVS Diagnosis: Tools and Service Aids.

GTF is most useful when you want to integrate trace entries from a CICS region with those from other CICS regions or other programs. GTF can record trace entries from all supported CICS releases, and it can be used by other programs besides CICS, such as the z/OS Communications Server (for SNA). You can relate CICS trace entries to those from the Communications Server using the task identifier in the trace header. However, because different products might run asynchronously, be cautious about using the sequence of trace entries in the GTF trace data set as evidence when you investigate problems.

To extract and format trace entries from GTF, you use the GTFTRACE subcommand in the interactive problem control system (IPCS). IPCS calls the CICS-supplied formatting routines DFHTG710 and DFHTR710. You can select the trace entries by specifying options on the USR parameter of the GTFTRACE subcommand for IPCS to select the trace entries, or by specifying options on the CICS parameter GTFTRACE subcommand for IPCS to pass to DFHTR710.

CICS exception tracing

CICS exception tracing is always done by CICS when it detects an exception condition. The sorts of exception that might be detected include bad parameters on a domain call, and any 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 at all are currently STARTED. That is why there is always an internal trace table in every CICS region, 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 main 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 MVS, 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 main 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 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 user exit programming interface (XPI) for programming information about XPI trace control function.

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 get a summary of its contents in a formatted CICS system dump when you specify 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 15 on page 103.

z/OS Communications Server exit tracing

z/OS Communications Server SNA exit tracing gives you a way of tracing SNA requests made from CICS.

You can control it online, using transaction CETR. See Figure 2 on page 72 for an illustration of the screen you need to use.

When CICS issues an SNA request, SNA services the request asynchronously and CICS continues executing. When SNA has finished with the request, it returns control to CICS by driving a CICS SNA exit. Every such exit contains a trace point, and if CICS SNA exit tracing is active, a trace entry is written to the GTF trace data set. GTF tracing must be active, but you do not need to start it explicitly from CICS. It is enough to start SNA exit tracing from the CETR transaction and terminal trace panel.

Note: The GTF trace data set can receive trace entries from a variety of jobs running in different address spaces. You need to identify the trace entries that have been made from the CICS region that interests you. You can do this by looking at the job name that precedes every trace entry in the formatted output.

You can use this type of tracing in any of the cases where you might want to use SNA buffer tracing, but it has the advantage of being part of CICS and, therefore, controllable from CICS. This means that you do not need a good understanding of SNA system programming to be able to use it. CICS SNA exit tracing also has the advantage of tracing some important CICS data areas relating to SNA requests, which might be useful for diagnosing problems.

Controlling CICS z/OS Communications Server exit tracing

You can turn CICS z/OS Communications Server SNA exit tracing on and off using the CETR transaction. You can specify tracing for just a single terminal, or for all the terminals in the SNA network. However, you cannot select which CICS exits are to be traced. Whenever CICS Communications Server exit tracing is running, you get a trace entry every time a CICS exit is driven by Communications Server.

If you select “normal” CICS tracing for the affected terminals at the same time as you have CICS Communications Server exit tracing running, you can then correlate CICS activities more easily with the asynchronous processing done by Communications Server.

If you must turn on CICS Communications Server exit tracing in an application owning region (AOR) while you are signed-on to a terminal in a terminal owning region (TOR), follow these steps:

1. Invoke CETR on the AOR.
2. Press PF5 to call up the CETR transaction and terminal trace screen.
3. Enter the APPLID of the TOR in the NETNAME field.
4. Complete other fields as required.
5. Press Enter.

CICS Communications Server trace entries are always written to the GTF trace data set, and you can format them in the usual way. See “Formatting and interpreting trace entries” on page 78 for more information. Direct all “normal” CICS tracing to the GTF trace destination as well, so you get the regular trace entries and the CICS Communications Server exit trace entries in sequence in a single data set. If you send the normal tracing to another destination, you get only the isolated traces from the exit modules with no idea of related CICS activity.

Interpreting CICS z/OS Communications Server exit trace entries

CICS z/OS Communications Server exit trace entries can be identified by their trace point IDs, which are in the range AP FC00 through AP FCFF. Not all the values in the range are used.

The format of the trace entries is similar to that shown in “Interpreting extended-format CICS system trace entries” on page 79. The interpretation string contains the netname of the terminal to which the trace entry relates, if the trace entry was made from a terminal-specific trace point. This makes it easy to identify the terminal associated with the Communications Server request. The trace entries also contain data from one or more selected CICS data fields, for example from the TCTTE. For guidance on interpreting the data values you might find there, see Trace entries overview.

z/OS Communications Server buffer tracing

z/OS Communications Server SNA buffer tracing enables you to look at all the data that is passed between logical units on an SNA communication link.

The trace entries, which include the netname of the terminal to which they relate, are made to the GTF trace data set. If you want to send “normal” CICS trace entries there, you can rationalize the activities of CICS with the asynchronous activities of SNA.

Selecting tracing by transaction

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.

About this task

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 "Setting trace destinations and tracing status" on page 77.

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 68.

When specific problems arise, you can set up special tracing so you can focus on just the relevant tasks and components. Use this procedure to specify the tracing you need:

Procedure

1. If you believe that specific tasks are involved in the problem, use special tracing:
   - When the problem is associated with a non-terminal task, or is associated with particular transactions, select special tracing for each suspect transaction.
   - When the problem is associated with particular terminals, select special tracing for each suspect terminal.
2. If you believe that specific components are implicated in the problem:
   a) For each suspected component, decide whether you need special level-1 tracing only, or level-1 and level-2 tracing.
   b) Turn special tracing off for all other components.
3. If you do not need standard tracing, turn the main system trace flag off.
4. Enable the trace destinations.

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 Table 7 on page 72.

Table 7. The combination of task trace options		
OPTION on TRANSACTION	**OPTION on TERMINAL**	**Task tracing**
tracing suppressed	**standard tracing**	SUPPRESSED
tracing suppressed	**special tracing**	SUPPRESSED
standard tracing	**standard tracing**	STANDARD
standard tracing	**special tracing**	SPECIAL
special tracing	**standard tracing**	SPECIAL
special tracing	**special tracing**	SPECIAL

You can set up the task tracing you want using the CETR transaction, with the screen shown in Figure 2 on page 72. You need to 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), and you can also select CICS z/OS Communications Server exit tracing for the terminal. For more details about CICS Communications Server exit tracing, see “z/OS Communications Server exit tracing” on page 69.

```
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 VTAM Exit Trace   ===>             ON, OFf
Terminal ZCP Trace         ===>             ON, OFf
VTAM Exit override         ===>  NONE       All, System, None
When finished, press ENTER.
PF1=Help      3=Quit     6=Cancel Exits      9=Error List
```

Figure 2. CETR screen for specifying standard and special task tracing

Note: VTAM® is now the z/OS Communications Server.

The CETR transaction can, for example, help you to get standard tracing for a transaction when it is run at one terminal, and special tracing when it is run at a second terminal.

Note:

1. You can turn standard tracing off for all tasks by setting the main system trace flag off. You can do this with the CETR transaction, using the screen shown in CETR - trace control, or you can code **SYSTR**=OFF at system initialization. However, any special task tracing will continueâ€”it is not affected by the setting of the system main 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 â€œComponents Trace Optionsâ€ screen shown in Figure 3 on page 74, 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 2 on page 72.

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 Table 8 on page 73. It is assumed that at least one trace destination is STARTED.

Table 8. Logic used to determine if a trace call is to be made from a trace point

Is tracing suppressed for this task?	Is standard tracing required for this task?	Is the main system trace flag on?	Is special tracing specified for this domain and trace level?	Is standard tracing specified for this domain and trace level?	Is trace call made?
Yes	not applicable	not applicable	not applicable	not applicable	No
No	Yes	Yes	not applicable	Yes	Yes
No	Yes	Yes	not applicable	No	No
No	Yes	No	not applicable	not applicable	No
No	No	not applicable	Yes	not applicable	Yes
No	No	not applicable	No	not applicable	No

Selecting tracing by component

You need to 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.

About this task

"Component names and abbreviations" on page 75 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 3 on page 74).

There are special considerations for the following CICS domains and their corresponding component codes:

AP (Application domain)
The component codes BF, BM, BR, CP, DC, DI, EC, EI, FC, IC, IS, KC, PC, RI, SC, SZ, TC, TD, TS, UE, WB, and WU are either entirely or partly 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.

MP (Managed platform domain)
In the Managed platform component (MP), one level of tracing, level 3, is intended for IBM field engineering staff. This trace level modifies the internal MP operation as follows:

MP level 3 trace
All updates to CICS Policy threshold counters are traced.

A significant performance overhead is introduced into your CICS system if this managed platform trace level is selected. Specifying MP =ALL activates MP trace levels 1, 2, and 3.

SM (Storage manager domain)
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. Specifying SM=ALL activates SM trace levels 1, 2, 3, and 4.

Defining component tracing at system initialization

You can code any of the following parameters to define component tracing at CICS system initialization time:

- SPCTR system initialization parameter, to indicate the level of special tracing required for CICS as a whole.
- SPCTR*xx*, 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).
- STNTR system initialization parameter, to indicate the level of standard tracing required for CICS as a whole.
- STNTRxx system initialization parameter, 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).

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 3 on page 74 shows you what the CETR Component Trace Options screen looks like. 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 3
Component Standard                          Special
--------  --------------------------------  ---------------------------
  AP      1-2                               1-2
  BA      1                                 1-2
  BM      1                                 1
  BR      1                                 1-2
  CP      1                                 1-2
  DC      1                                 1
  DD      1                                 1-2
  DH      ALL                               1-2
  DM      1                                 1-2
  DP      1                                 1-2
  DS      1                                 1-2
  DU      1                                 1-2
  EC      1                                 1-2
  EI      1-2                               1-2
  EJ      1                                 1-2
  EM      1                                 1-2
  EP      1                                 1-2
  FC      1                                 1-2
  GC      1                                 1-2
  IC      1                                 1-2
  IS      1                                 1-2
PF:  1=Help   3=Quit   7=Back  8=Forward  9=Messages   ENTER=Change
```

Figure 3. CETR screen for specifying component trace options

The trace levels for a specific CICS component are represented by two values. One value shows the active level of tracing for standard tracing; the other shows possible levels for special tracing. In CETR, you can set the active level of tracing for standard or special tracing for an individual component or for a group of components.

In this example, all components except AP, DH and EI have level-1 tracing enabled during standard tracing. AP and EI have both level-1 and level-2 tracing enabled, which will provide more detailed information than level-1 tracing alone. The DH component has a standard trace setting of **ALL**, which will provide the maximum amount of trace information possible.

For special tracing, all components except BM and DC can have level-1 and level-2 tracing enabled.

Level-1 trace points are designed to give you enough diagnostic information to fix errors caused by user applications or user actions, while level-2 trace points provide information that is likely to be more useful for fixing errors within CICS code. Generally, you will not want to use level-2 trace points yourself, unless requested to do so by IBM support staff after you have referred a problem to them.

Component names and abbreviations

CICS components are abbreviated to a 2-letter code to make interfaces, such as the CETR transaction, easier to use.

Code	Component name
AP	Application domain
AS	Asynchronous services domain
BA	Business application manager
BM*	Basic mapping support
BR*	3270 bridge
CP*	Common programming interface
DC*	Dump compatibility layer
DD	Directory manager domain
DH	Document handling domain
DM	Domain manager domain
DP	Debugging profiles domain
DS	Dispatcher domain
DU	Dump domain
EC*	Event capture and emission
EI*	Exec interface
EJ	Enterprise Java domain
EM	Event manager domain
EP	Event processing domain
FC*	File control
GC	Global catalog domain
IC*	Interval control
IE	ECI over TCP/IP domain
IS*	ISC or IRC
KC*	Task control
KE	Kernel
LC	Local catalog domain
LD	Loader domain
LG	Log manager domain

Code	Component name
LM	Lock domain
ME	Message domain
ML	Markup language domain
MN	Monitoring domain
MP	Managed platform domain
NQ	Enqueue domain
OT	Object transaction domain
PA	Parameter domain
PC*	Program control
PG	Program manager domain
PI	Pipeline domain
PT	Partner domain
RA	Resource manager adapters
RI*	Resource manager interface (RMI)
RL	Resource life-cycle domain
RM	Recovery manager domain
RS	Region status domain
RX	RRS-coordinated EXCI domain
RZ	Request streams domain
SC*	Storage control
SH	Scheduler services domain
SJ	JVM domain
SM	Storage manager domain
SO	Sockets domain
ST	Statistics domain
SZ*	Front End Programming Interface (FEPI)
TC*	Terminal control
TD*	Transient data
TI	Timer domain
TR	Trace domain
TS	Temporary storage domain
UE*	User exit interface
US	User domain
WB	Web domain
WU	CICS Management Client Interface (CMCI) domain

Code	Component name
W2	Web 2.0 domain
XM	Transaction manager domain
XS	Security manager domain

Notes:

1. Components marked ***** are subcomponents of the AP domain. The trace entries for these components are produced with a trace point ID of AP *nnnn*.
2. For the DS domain function CHANGE_MODE, a trace entry is generated if DS level 2 or 3 tracing is active.

Setting trace destinations and tracing status

You can set the system tracing status by using the appropriate system initialization parameters, or by using the CETR transaction or the **SET TRACETYPE** command when CICS is running. You can use the CETR transaction and **SET TRACETYPE** command to make changes in response to contingencies as they arise.

About this task

There are four possible destinations for trace entries in CICS:

- The internal trace table
- The auxiliary trace data sets
- The MVS generalized trace facility (GTF) data sets
- The JVM server trace file in z/OS Unix System Services

You can select any combination of tracing, based on these factors:

- The characteristics of the various types of CICS tracing
- The amount of trace data that you need to capture
- Whether you want to integrate CICS tracing with tracing done by other programs

For information about the different trace destinations, see “Trace destinations” on page 66.

Procedure

- To set up the tracing status at system initialization, use the following system initialization parameters:
  - AUXTR, to specify whether auxiliary trace is to be on or off at CICS startup.
  - AUXTRSW, to specify whether or not automatic switching takes place for auxiliary trace data sets when they are full.
  - GTFTR, to specify whether CICS is to use GTF as a destination for CICS trace data.
  - INTTR, to specify whether internal tracing is to be on or off at CICS startup.
  - SYSTR, to set the main system trace flag on or off at CICS startup.
  - TRTABSZ, to specify the size of the internal trace table.
  - TRTRANSZ, to specify the size of the transaction dump trace table, which is the copy of the internal trace table that CICS makes in the event of a transaction dump.
  - USERTR, to set the main user trace flag on or off at CICS startup. This flag must be on if your applications make user trace calls.
- To set the tracing status while CICS is running, use the CETR transaction or the CEMT transaction:

  a) If you want to use standard tracing or capture user trace entries from applications, ensure that the main system trace flag is set to ON.

If it is OFF, no standard tracing is done at all, even though standard tracing might be specified for some tasks. Also, any trace call requests in your programs are ignored. You can see the role of the main system trace flag in Logic used to determine if a trace call is to be made from a trace point.

b) If you want to direct regular tracing explicitly to the internal trace table, set internal tracing status to STARTED.

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. It is also used as the destination for exception trace entries.

c) To use GTF tracing, set the GTF trace status to STARTED. Ensure that the GTF trace data set is defined to MVS.

Be aware that no error condition is reported if the CICS GTF status is started but GTF tracing has not been started under MVS. If this happens, the trace entries are not written. To write trace entries, MVS GTF trace must be started with the TRACE=USR option before CICS GTF trace is started.

d) To start writing entries to the auxiliary trace data sets, set the auxiliary trace status to STARTED.

If you have two auxiliary trace data sets, you can use the auxiliary switch to specify the action CICS takes when one data set is full. For an explanation of the actions, see "Auxiliary trace data sets" on page 67.

e) To start writing JVM server tracing, use the CETR transaction to trace the SJ and AP components. JVM servers do not use auxiliary or GTF tracing. Instead, a small amount of trace is written to the internal trace table and the rest of the trace is written out to a file in zFS that is unique for the JVM server. For more information, see Activating and managing tracing for JVM servers in Troubleshooting and support.

f) To stop internal tracing, GTF tracing, or auxiliary tracing, set their status to STOPPED.

For auxiliary tracing, you can also set a status of PAUSED. With this status, CICS stops writing entries to the auxiliary trace data set, but leaves the data set open.

- Alternatively, to set the tracing status while CICS is running, use the SET TRACETYPE command.
- To change the size of the internal trace table while CICS is running, use the CETR transaction.

  When you change the size of the internal trace table, 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.

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 do some formatting. The way you do the formatting varies depending on the destination.

You can specify **abbreviated**, **short**, or **full** trace formatting, to give you varying levels of information and detail in your output. Typically, abbreviated-format trace gives you one line of trace per entry; short-format provides two lines of trace per entry; full-format provides many lines of trace per entry. The structures of the different types of trace entry are described in the sections that follow.

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, in error situations, you might require more information than the abbreviated trace can provide. The short trace provides the information that is presented in the abbreviated trace, and, additionally, presents certain items that are presented in the full trace. These are:

- Interpreted parameter list
- Return address
- Time that the trace entry was written
- Time interval between trace entries

These items of information are often very useful in the diagnosis of problems. By selecting the short format, you can gain access to this information without having to bear the processing overhead of formatting a full trace, and without having to deal with the mass of information in a full trace.

There may be occasions, however, when you need to look at full format trace entries, to understand more fully the information given in the corresponding abbreviated and short entries, and to be aware of the additional data supplied with many full trace entries.

For abbreviated and full trace formatting, a trace summary table provides summary information about the trace entries that relate to each task in the system during the time period. The trace summary table appears at the end of the formatted trace output. Use the table to see the tasks that were traced, and the location and number of their trace entries in the trace output. The table also highlights any long time gaps between the trace entries for a task, which can indicate a performance problem, and any exception trace entries for a task. The trace summary table is not produced for short-format trace.

The internal trace table can be formatted in one of two ways:

1. From a CICS system dump, using the CICS print dump exit, DFHPD710.
2. From a transaction dump, using the CICS dump utility program, DFHDU710.

Auxiliary trace can be formatted using the CICS trace utility program, DFHTU710. 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.

Note: Trace entries can only be formatted selectively by transaction or terminal if the “transaction attach” entry (point ID XM 1102, XM level-1) for the transaction is included in the trace data set. The transaction attach entry will not be written if for example the main system trace flag is switched off and the transaction status is set to special.

GTF trace can be formatted with the same sort of selectivity as auxiliary trace, using a CICS-supplied routine with the MVS interactive problem control system (IPCS).

For more details of trace utility programs, see Trace utility print program (DFHTU700).

Interpreting extended-format CICS system trace entries

CICS system trace entries made to the internal trace table, the auxiliary trace data sets, and the GTF trace data set can all be formatted to give the same type of information.

About this task

There are two slightly different extended trace entry formats. The short style resembles the format used in earlier releases of CICS, and gives FIELD A and FIELD B values. The other long style uses a different format, described below.

Procedure

1. Look at 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 00E1         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
   ```

2. Look at the interpretation string.

It shows:

- 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.

3. Look at the standard information string. It shows:
   - 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. (Note that the GTF trace time is GMT time.)
   - 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 CICS TCB ID and the address of the **MVS TCB** (field TCB) that is in use for this task. This field can help you in comparing a CICS trace with the corresponding MVS trace. As there can be multiple OTE TCBs, the TCB ID for an OTE TCB is in the format *ccnnn* where *cc* identifies the type of OTE TCB (e.g. X9, L8 etc.) and *nnn* is a sequence number identifying which of the OTE TCBs is in use.
   - 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.
4. Read the **data fields**, that contain information relevant to the function being performed.

   For short 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. Some short 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.

   For long 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 trace.

   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.

Examples of the extended format for short and long trace entries

The tracing in the AP domain has two different styles of trace entry; the oldest style of trace entry is short and contains minimal information, whereas the later style of trace entry is long, containing much more information to help with problem determination.

Figure 4 on page 81 shows a trace entry made from a short trace point. Its trace point ID is AP 00E1, corresponding to trace ID X'E1' in old releases of CICS.

```
AP 00E1 EIP EXIT INQUIRE-PROGRAM OK        REQ(00F4) FIELD-A(00000000 ....) FIELD-B(00004E02 ..+.) BOUNDARY(0200)
           TASK-00048 KE_NUM-07FC TCB-L8000/009A0E88 RET-B28102A2 TIME-11:19:53.2944533850 INTERVAL-00.0000226252   =000491=
```

Figure 4. Example of the extended format for a short trace entry

Note: For some trace entries, an 8-character resource field is shown after FIELD B.

The explanation of the short trace entry is as follows:

- AP 00E1 shows that this trace entry was made from trace point X'00E1' in the application domain.

  Although all short trace points are in AP domain, not all AP domain trace points are short. Some are long trace points and they have a similar format to that shown in Figure 5 on page 81. In general, short 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'
- EIP EXIT INQUIRE-PROGRAM OK is the interpretation string, which 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.
  - INQUIRE-PROGRAM shows the type of function requested.
- REQ(00F4) represents the request type of the short trace format. 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 short 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'4E02', showing that the EXEC CICS command was **INQUIRE PROGRAM**.

  Note: For some trace entries, an 8-character resource field is shown after FIELD B.
- BOUNDARY shows the dispatcher state and the EIS boundary flags.
- The standard information string shows:
  - TASK-00048 shows that the task number for the currently running task is 00048. Any trace entries having the same task number would have been made while this task was running.
  - KE_NUM-07FC shows that the kernel task number for the task is 07FC. If you took a system dump while this task was in the system, you could identify the task in the kernel summary information from this number.
  - TCB-L8000/009A0E88 is the address of the MVS TCB.
  - RET-B28102A2 is the return address.
  - The time when the trace entry was made was 11:19:53.2944533850.
  - The interval that elapsed between this and the preceding trace entry was 00.0000226252 seconds.

Figure 5 on page 81 shows a long trace entry.

```
SM 0C01 SMMG  ENTRY - FUNCTION(GETMAIN) GET_LENGTH(1A4A) SUSPEND(YES) INITIAL_IMAGE(00) STORAGE_CLASS(TASK)
              TASK-00163 KE_NUM-0007 TCB-QR   /009F3338 RET-800411F2 TIME-16:31:52.5916976250 INTERVAL-00.0000666250    =000112=
                1-0000  00480000 00000011 00000000 00000000  B6700000 00000000 02000100 C4C6C8C3
*.........................DFHC*
                0020  C5E3D9C4 03BD5BB0 00001A4A 03BD5B01  00000001 01000698 04755D70 40400008
*ETRD..$.... ..$........Q..).  ..*
                0040  00000FE8 C3C5E3D9
*...YCETR                     *
```

Figure 5. Example of the extended format for a long trace entry

The explanation of the long trace entry is as follows:

- SM 0C01 shows that this trace entry was made from trace point X'0C01' in the storage manager domain.

- `SMMG ENTRY - FUNCTION(GETMAIN) GET_LENGTH(1A4A) SUSPEND(YES) INITIAL_IMAGE(00) STORAGE_CLASS(TASK)` is the interpretation string, which provides the following information:
  - SMMG tells you that the trace call was made from module DFHSMMG.
  - ENTRY FUNCTION(GETMAIN) tells you that 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 provides the following 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 following the standard information was taken from only one data area.

  Storage manager domain trace points 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.

  Information about some data areas is intended for use by IBM support personnel, and therefore the 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, contact your IBM Support Center.

Interpreting short-format CICS system trace entries

About this task

Short-format trace entries contain the information that is presented in the abbreviated-format trace entry and the following items from the interpretation string of the extended-format trace entry:

- Interpreted parameter list, showing keyword and value
- Return address
- Time
- Interval

Procedure

- If you are using the short-format for an old-style trace entry, use the following example to help you interpret the trace.
  Figure 6 on page 82 shows an example of the short-format for an old-style entry.

```
00030 QR AP 00E1 EIP ENTRY INQUIRE-TRACEFLAG REQ(0004) FIELD-A(071F6018 ..-.) FIELD-B(08007812 ....)
                                                  RET-870844CE 11:39:44.8516351250 00.0000343750 =000011=
```

Figure 6. Example of the short-format for an old-style trace entry

In this example:

- `00030` is the task number
- `QR` is the TCB ID

- AP 00E1 is the trace point ID
- EIP ENTRY INQUIRE-TRACEFLAG is the interpretation string
- REQ(0004) is the type of request
- FIELD-A(071F6018 ..-.) FIELD-B(08007812 ....) are the values of FIELDA and FIELDB respectively, with their EBCDIC interpretation.
- RET-870844CE is the CALL return address
- 11:39:44.8516351250 is the time the trace entry was made
- 00.0000343750 is the interval since the last trace entry
- =000011= is the trace entry number

- If you are using the short-format for a new-style trace entry, use the following example to help you interpret the trace.
  Figure 7 on page 83 shows an example of the short-format for a new-style trace entry.

```
035925 QR SM 0C01 SMMG ENTRY GETMAIN GET_LENGTH(6A80SUSPEND(NO) INITIAL_IMAGE(00) STORAGE_CLASS(USER24) CALLER(EXEC)
                                                    RET-8735C8AC 16:28:40.7980146252 00.0000308750 =000013=
```

Figure 7. Example of the short-format for a new-style trace entry

In this example:

- 035925 is the task number
- QR is the TCB ID
- SM 0C01 is the trace point ID.
- SMMG ENTRY GETMAIN GET_LENGTH(6A80SUSPEND(NO) INITIAL_IMAGE(00) STORAGE_CLASS(USER24) CALLER(EXEC) is the interpretation string, including the interpreted parameter list.
- RET-8735C8AC is the CALL return address
- 16:28:40.7980146252 is the time the trace entry was made
- 00.0000308750 is the interval since the last trace entry
- =000013= is the trace entry number

The following example denotes the tracing of different OTE TCBs. Notice in particular that the OTE TCB IDs have a sequential number associated with them to indicate the TCB that is in use.

```
00258 X90A4 SM 0301 SMGF  ENTRY GETMAIN SUBPOOL_TOKEN(27E5AAAC , 0000007F) GET_LENGTH(448) SUSPEND(YES)
INITIAL_IMAGE(00) REMARK
                                                          (APPIS)  RET-9BAC3384 14:25:09.6470453803
00.0000002812 =002602=
00258 X90A4 SM 0302 SMGF  EXIT  GETMAIN/OK ADDRESS(1C790000)               RET-9BAC3384 14:25:09.6470490678
00.0000036875 =002603=
00258 X90A4 SM 0301 SMGF  ENTRY GETMAIN SUBPOOL_TOKEN(27E5AC14 , 00000081) GET_LENGTH(104) SUSPEND(YES)
INITIAL_IMAGE(00) REMARK
                                                          (APPIS)  RET-9BAC3412 14:25:09.6470493803
00.0000003125 =002604=
00258 X90A4 SM 0302 SMGF  EXIT  GETMAIN/OK ADDRESS(000C0320)               RET-9BAC3412 14:25:09.6470503803
00.0000010000 =002605=
00259 X90A6 DS 0010 DSBR  ENTRY INQUIRE_TCB                                RET-9BABF518 14:25:09.6470622773
00.0000118969 =002606=
00259 X90A6 DS 0011 DSBR  EXIT  INQUIRE_TCB/OK OWNER_TCB_TOKEN(1C387BE0)   RET-9BABF518 14:25:09.6470658085
00.0000035312 =002607=
```

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.

About this task

Abbreviated trace entries show the CICS TCB ID of the TCB instead of an MVS TCB address.

Procedure

- If you are using old-style trace entries, use the following example to help you interpret the trace.

```
00021 QR    AP 00E1 EIP   ENTRY INQUIRE-TRACEFLAG        0004,00223810 ....,00007812 ....         =000005=
```

Figure 8. Example of the abbreviated format for an old-style trace entry

  In this example:

  – 00021 is the task number
  – QR is the TCB ID
  – AP 00E1 is the trace point ID
  – EIP ENTRY INQUIRE-TRACEFLAG is the abbreviated interpretation string
  – 0004 is the request field
  – 00223810 .... is FIELDA, with its EBCDIC interpretation
  – 00007812 .... is FIELDB, with its EBCDIC interpretation
  – =000005= is the trace entry number

  Note: For some trace entries, an 8-character resource field appears to the right of FIELD B. Also, some trace entries 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.

- If you are using new-style trace entries, use the following example to help you interpret the trace.

```
00021 QR    LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK    03B8A370 , 00000001,848659C0,048659A0,410,200,REUSABLE   =000023=
```

Figure 9. Example of the abbreviated format for a new-style trace entry

  In this example:

  – 00021 is the task number
  – QR is the TCB ID
  – AP 00E1 is the trace point ID
  – LDLD EXIT ACQUIRE_PROGRAM/OK 03B8A370 , 00000001,848659C0,048659A0,410,200,REUSABLE is the abbreviated interpretation string
  – =000005= is the trace entry number

  a) 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 (or short-format) trace entry to find out what they are. Figure 10 on page 84 shows the corresponding extended-format trace entry.

```
LD 0002 LDLD EXIT - FUNCTION(ACQUIRE_PROGRAM) RESPONSE(OK) NEW_PROGRAM_TOKEN(03B8A370 , 00000001) ENTRY_POINT(848659C0) LOAD_POINT
         (048659A0) PROGRAM_LENGTH(410) FETCH_TIME(200) PROGRAM_ATTRIBUTE(REUSABLE)
       TASK-00021 KE_NUM-0007 TCB-QR   /009FF3C0 RET-847B26A2 TIME-10:45:49.6888118129 INTERVAL-00.0000235625    =000023=
         1-0000  00880000 0000001C 00000000 00000000  BBA02800 00000000 01000100 C4C6C8C3  *.h.........................DFHC*
           0020  D9D84040 FD052000 00062060 03B8A370  00000001 848659C0 048659A0 A4F78696  *RQ  .......-..t.....df...f..u7fo*
           0040  00000410 C3D9E2D8 00000000 C3C9C3E2  E4E2C5D9 01010002 1C000000 00000000  *....CRSQ....CICSUSER...........*
           0060  00000000 00000200 C302D840 40000500  01000000 00000000 00000000 00000000  *........C.Q  ...................*
           0080  00000000 00000000                                                        *........                       *
```

Figure 10. 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. (Note that the GTF trace time is GMT time.)
– 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 Lock manager domain trace points in IBM Knowledge Center for details of trace point ID LD 0002, you will see that the data area is the LDLD parameter list.

The following example denotes the tracing of different TCBs. Notice in particular that the OTE TCB IDs have a sequential number associated with them to indicate the TCB that is in use.

```
00255 QR    SM 0D01 SMMF  ENTRY FREEMAIN              EIIC TEM,1D710958,TEMPSTG
=001561=
00256 L9016 SM 0301 SMGF  ENTRY GETMAIN               CAD0,YES,LE_RUWA,TASK31
=001562=
00255 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK           USER storage at 1D710958
=001563=
00256 L9016 LM 0003 LMLM  ENTRY LOCK                  1AF55C48,EXCLUSIVE
=001564=
00256 L9016 LM 0004 LMLM  EXIT  LOCK/OK
=001565=
00255 L9015 AP 00E1 EIP   EXIT  RETRIEVE              OK                          00F4,00000000 ....,0000100A ....
=001566=
```

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.

About this task

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 11 on page 85.

```
AP 000B USER  EVENT - APPLICATION-PROGRAM-ENTRY - SEND     - CICS USER TRACE ENTRY           HELP INFORMATION
              TASK-00163 KE_NUM-0007 TCB-QR   /009F3338 RET-8003F54C TIME-16:32:01.1295568750 INTERVAL-00.0001965625    =000731=
                1-0000  E4E2C5D9 40404040                                                    *USER                          *
                2-0000  C3C9C3E2 40E4E2C5 D940E3D9 C1C3C540  C5D5E3D9 E8404040 40404040 40404040  *CICS USER TRACE ENTRY         *
                  0020  C8C5D3D7 40C9D5C6 D6D9D4C1 E3C9D6D5  40404040 40404040 40404040 40404040  *HELP INFORMATION              *
                  0040  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                              *
                  0060  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  *                              *
                3-0000  E2C5D5C4 40404040                                                    *SEND                          *
```

Figure 11. 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 11 on page 85 is shown in Figure 12 on page 86.

```
00163 QR    AP 000B USER  EVENT APPLICATION-PROGRAMRY SEND CICS USER TRACE ENTRY HELP INFORMATION      =000731=
```

Figure 12. 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 see the corresponding extended trace entries to assess their significance. Figure 13 on page 86 shows an example of the short format for a user trace entry.

```
00031 QR    AP 000B USER  EVENT APPLICATION-PROGRAM-E SEND  - CICS USE RET-800820A2 11:42:27.1176805000 00.0000 247500 =00 0815=
```

Figure 13. Example of the short format for a user trace entry

Chapter 6. Dealing with the problem

Following on from classifying your problem, this section describes how to find the cause of problems in each area.

Dealing with transaction abend codes

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 section to record your findings (“Worksheet for transaction abends” on page 96).

Collecting the evidence

The evidence you need to investigate the transaction abend should be in the information sent to the various transient data queues for error messages, and in the transaction dump.

Procedure

1. Check the transaction dump to find information about the transaction abend.

   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 can use the symptom string to search the RETAIN database for any similar problems. For an introduction to symptom strings and their contents, see “Symptom strings” on page 18.

   If no transaction dump has been produced, it is possible that transaction dumping has been suppressed for the transaction (through 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 “Using dumps in problem determination” on page 18.
2. Check the CSMT log.

   The transaction abend code and the abend message are recorded in the log. Make a note of any other messages in the log that might relate to the abend, because they might provide additional valuable evidence.
3. Check whether any relevant messages were sent to the transient data destinations used by CICS to record messages.

   Look in particular for any messages about files, terminals, or printers that you might be attempting to use.

   For a list of destinations used by CICS, see Setting up data sets for transient data.

What the abend code can tell you

The first thing that the transaction 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 detailed information and a full list of the transaction abend codes used by CICS and by other IBM products, see CICS messages.

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.

Transaction abend codes: AEYD, AICA, ASRA, ASRB, and ASRD

Special procedures apply to the AEYD, AICA, ASRA, ASRB, and ASRD abend codes.

For an abend code other than AEYD, AICA, ASRA, ASRB, and ASRD, use the procedures in “Last statement identification” on page 59 to find the last command that was executed, and then see “Analyzing the problem further” on page 96. For details about CICS abend codes, see Transaction abend codes. All transaction abend codes that CICS issues are listed, with an explanation of why the code was issued, and details of system and user actions. The same information is available online, using the CICS-supplied messages and codes transaction, CMAC.

If, after you review this information, you cannot find the cause of the problem, continue with the procedures in “Dealing with transaction abend codes” on page 87.

AEYD abends

If command protection is activated by the CMDPROT(YES) option in the system initialization table (SIT), 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 that contains the invalid address. The trace should include an exception trace entry that is created by DFHEISR or DFHEIGR and that identifies the parameter in error. If the abend is handled, EXEC CICS ASSIGN ASRASTG, ASRAKEY, and ASRASPC can give additional information about the abend.

To prevent a recurrence of the abend, correct the program code. Alternatively, changing one or more of the following options might alleviate the problem:

- EXECKEY in the program definition, if storage protection is active
- TASKDATAKEY in the transaction definition
- ISOLATE in the transaction definition, if transaction isolation is enabled

For further information, see “Avoiding storage violations” on page 231.

AICA abends

If your transaction terminated with abend code AICA, the transaction was probably in a loop. For detailed guidance about dealing with loops, see “Dealing with loops” on page 196.

ASRA abends

CICS issues an ASRA abend code when it detects that a program check has occurred in 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 machine hardware uses the PSW 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.

ASRB abends

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

Use the procedures in "Locating the last command or statement" on page 58 to find the origin of the abend in your program. Use that information, and the description and procedures for ASRB abends in Transaction abend codes, to resolve the problem.

ASRD abends

A transaction abends with code ASRD in the following situations:

- An application program attempts to invoke CICS macros.
- An application program attempts to access the common service area (CSA) or task control area (TCA).

These situations cause 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.

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.

About this task

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 will have preceded the abend
- The transaction abend control block (TACB) which will have been created to describe the abend
- Exception trace point ID AP 0781 for an ASRA abend or AP 0783 for an ASRD abend.

See "Interpreting transaction dumps" on page 55.

Procedure

1. Find the offset of the program check within the program load module.

   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.

   - If the offset is not X'FFFFFFFF', go to "What type of program check occurred?" on page 90.
   - If the offset is X'FFFFFFFF', continue following the steps.
2. When the offset is X'FFFFFFFF', CICS was unable to establish the location of the program check. Use the PSW to obtain the next sequential instruction address.

   The PSW can 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

3. 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 58 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 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.

4. 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 96.

What type of program check occurred?

Knowing what type of program check occurred can be helpful to find the cause of the error. The type of program check is indicated by the program interrupt code (PIC), which you can find in the program status word (PSW) at the start of the transaction dump.

For details about the PSW, see z/Architecture Principles of Operation.

PIC
PIC explanation

1

Operation exception - incorrect operation attempted.

Possible causes are as follows:

- Overlaid program
- Overlaid register save area, causing incorrect branch
- Resource unavailable, but program logic assumed valid address 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.

A possible cause is as follows:

- Incorrect branch to this code. Possible reasons for this situation are as follows:
  - 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 are as follows:

- Incorrect branch to this code
- Incorrect register contents. Possible reasons for this situation are as follows:
  - 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 are as follows:

- Resource unavailable, and return code not checked. Program logic assumed valid address returned and took inappropriate action.
- Incorrect interface parameters to another 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 use 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, RDSA, ECDSA, ERDSA, ETDSA, or GCDSA.
- Attempt to write to the ERDSA or RDSA when PROTECT is specified for the **RENTPGM** parameter.
- Attempt to read or write to another transaction's storage. For example, in a system running with transaction isolation, a program running in USER key might experience a protection exception when attempting to access the USER key task-lifetime storage of another transaction.
- Storage that is 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.

A possible cause is as follows:

- Incorrect register contents, which might be because of an overlaid register save area.

6

Specification exception - incorrect format of an instruction or invalid registers.

Possible causes are as follows:

- 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 both of the operands contain data that is not suitable for the instruction.

Possible causes are as follows:

- Incorrect input data (often because blanks are 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 are as follows:

- 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

About this task

If the program check was due to an arithmetic error (interruption codes 7 through F), you need to find the operands used in the last instruction.

Procedure

1. Use the procedure described in section “Locating program data” on page 59 to locate the fields.
2. Check that the operands are valid.

   You need to know a little about the type of arithmetic being done, so that you can verify the operands are correct. 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.
3. 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

Storage protection, transaction isolation, and command protection are facilities that add data integrity by highlighting application errors. The use of these facilities greatly reduces the number of abends that appear to be CICS problems.

About this task

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

- An attempt is made to write to the CDSA, RDSA, ECDSA, ERDSA, or GCDSA 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 transaction isolation (for which storage protection is a prerequisite) is enabled, additional situations can occur:

- A transaction, defined with ISOLATE(YES), is executing a USER key program and attempts to read or write to another transaction's USER key task-lifetime storage in the UDSA or EUDSA.
- A transaction, defined with ISOLATE(NO), is executing a USER key program and attempts to read or write to another transaction's USER key task-lifetime storage in the UDSA or EUDSA, but the second transaction is defined with ISOLATE(YES). For a full description of the transaction isolation facility and its use, see TRANSACTION attributes.

If any of these situations occur, 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, and whether the program was executing in a subspace (CDSA, UDSA, RDSA, ECDSA, EUDSA, ERDSA, ETDSA, GCDSA, or GUDSA). This information is 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, a protection exception can occur if storage that is passed to CICS as an output parameter through the EXEC interface is not accessible for READ/WRITE by the program that issued the command. The program is passing to CICS storage that it cannot itself update, but it requires CICS to update the storage. 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.

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 run in an upgraded 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, and, as a result, loaded by CICS into one of the read-only DSAs (RDSA, 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.

Transaction isolation

Transaction isolation protects the data associated with a user transaction from being overwritten by EXECKEY(USER) programs invoked by other user transactions.

If transaction isolation is active, the occurrence of a protection exception in a new transaction indicates a probable error in the transaction or program definition. An interdependency might exist between two or more transactions. In a system running without transaction isolation, a transaction can read or write to the task-lifetime storage of another transaction. The CICS Interdependency Analyzer helps to identify potential dependencies. Ideally, such interdependencies should be removed. If interdependencies cannot be removed, define all affected transactions with ISOLATE(NO).

For further details about defining transactions, see TRANSACTION attributes. For more information about CICS Interdependency Analyzer, see CICS Interdependency Analyzer for z/OS.

Command protection

Command protection 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 upgrading 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 list summarizes some of the causes of protection exceptions that can occur in user key programs

- Issuing an MVS macro request. Most MVS macros and services are not supported in EXECKEY(USER) application programs. Use of unsupported macros and services might cause a failure if these macros or services attempt to reference MVS storage outside the CICS DSAs.
- Referencing storage that is obtained by an MVS GETMAIN request or another MVS macro. MVS storage that is 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 Developing applications 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, might also cause MVS storage outside the CICS DSAs to be obtained or updated (such storage is protected from user-key programs).

  In previous releases of CICS, these might have worked, or at least might 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 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.

  Note: If you are using CSP/AD, CSP/AE, or CSP/RS, you must ensure that the definitions for programs DCBINIT, DCBMODS, DCBRINIT and DCBNCOP specify EXECKEY(CICS). These are all examples of programs that modify global work areas that are set up by global user exit programs.

- If you are using DB2® and you use the DB2 message formatting routine DSNTIAR, which is link-edited with your application programs, you should apply the PTF for DB2 APAR PN12516, and relink-edit the applications using DSNTIAR so that they can run in user key. If the applications are not re-link-edited after this PTF is applied, they will have to run in CICS key. As a first step, until you have applied this PTF, you can define the applications which use DSNTIAR with EXECKEY(CICS).

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 rather than making the program non-reentrant.

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 RENT specified, using a variable in the program 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 RENT attribute and then attempting to initialize or update the table during CICS execution. Such a table should not be defined as RENT.
- Defining BMS mapsets as RENT 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 RENT attribute. 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 RENT attribute. (Partition sets are not modified by CICS and can be link-edited with the RENT attribute.)

Protection exceptions referencing the UDSA and EUDSA

In a system running with transaction isolation enabled, protection exceptions can occur in programs with EXECKEY(USER).

Such an exception is caused by one transaction using a user key program to read or write to the user-key task-lifetime storage of another transaction. This situation might highlight a program error or an interdependency between two transactions. The IBM CICS Interdependency Analyzer for z/OS tool can help to identify potential transaction interdependencies. Examples of transaction interdependencies are:

- A transaction might use EXEC CICS GETMAIN or GETMAIN64 to obtain 24-bit or 31-bit storage, and pass the address of the storage to other transactions. Access to this storage by one of these other transactions causes a protection exception if transaction isolation is enabled, unless both affected transactions are defined with ISOLATE(NO). Storage to be shared in this manner should be acquired by a GETMAIN with the SHARED option. This is preferable to defining the transactions with ISOLATE(NO).
- A transaction might attempt to post an ECB that exists in another transaction's task-lifetime storage. ECBs should be acquired by a GETMAIN from shared storage. Alternatively, the affected transactions should be defined with ISOLATE(NO).

Transaction isolation does not apply to 64-bit storage, so protection exceptions caused in this way do not reference the GUDSA.

Analyzing the problem further

About this task

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.

Abends when CICS is using the DBCTL interface

If a transaction terminates abnormally while CICS is using DBCTL, you need to determine whether CICS or IMS was in control at the time of the abend.

You can do this by examining the time stamps in the CICS and DBCTL traces. For guidance about this, see ../dfht443.dita#dfht443.

If tracing was off at the time of the failure, you can find an indicator in the task local work area for DFHDBAT. The indicator is set when CICS passes control to DBCTL, and reset when DBCTL returns control to CICS.

To find the indicator, locate the eye-catcher for the TIE in the dump and then locate the LOCLAREA eye-catcher that follows it. The indicator is at offset `X'14'` from the start of the LOCLAREA eye-catcher. If the indicator byte is set to `X'08'`, CICS has passed control to DBCTL, and you should examine the IMS part of the transaction. If the byte is set to `X'00'`, DBCTL has returned control to CICS, and you should investigate the CICS part of the transaction.

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?**
   - If this is a USER abend code, tell the appropriate person.
   - For a CICS abend code, continue with “3” on page 96.
3. **Look up the abend code**

   If you need further advice, continue with “4” on page 96.
4. **Is this an AICA abend?**

   If it is, read “Dealing with loops” on page 196. If not, continue with “5” on page 96.
5. **Is this an ASRA abend?**

   If it is, go to step “7” on page 97. If not, continue with “6” on page 96.
6. **Is this an ASRD abend?**

   If it is, continue with “7” on page 97. If not, go to “14” on page 97.

7. **Record the program areas from the dump.**

   Find the program names from the Module Index at the end of the formatted dump. For each program, record the program name, the beginning address, and end address.
8. **Record the address of the next instruction from the PSW, or the offset established by CICS.**
9. **Did the program check occur in one of the program areas listed above?**

   If it did, continue with "10" on page 97. If not, go to "14" on page 97.
10. **Record what type of program check occurred.**

    You will need to record the Program Interrupt Code (PIC).
11. **Find the last statement executed.**

    See "Locating the last command or statement" on page 58.
12. **Was the PIC one of the arithmetic interrupts (7,8,9,A,B,C,D,E,F)?**

    If it was, find the contents of the operands of the last instruction (see "Locating program data" on page 59), and go to step "15" on page 97. If not, continue with "13" on page 97
13. **Was the PIC a protection exception?**

    If it was, read "Dealing with protection exceptions" on page 93.

    Go to "15" on page 97.
14. **Find the last statement executed**

    See "Locating the last command or statement" on page 58.
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:

    - Parameters to or from other programs or systems.
    - Any needed resource that may not be available.
    - The formatted trace, for any unexplained flow.
    - The running environment, for any changes in it.

FEPI abends

For information about FEPI-associated abends in CICS or MVS, see FEPI abends.

Dealing with CICS system abends

This information provides guidance about gathering essential information about CICS system abends.

If you have not done so already, use the CMAC transaction or refer to CICS messages for an explanation of any message you have received, because it could offer a straightforward solution to your problem. For further information about the CMAC transaction, see CMAC - messages and codes display.

If the abend was clearly caused by a storage violation, refer to the information in "Dealing with storage violations" on page 231. You know when CICS has detected a storage violation, because it issues the following message:

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

After you refer to the information about storage violations, if you find the cause of the abend is an application error, you must investigate 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 you do so, you must gather the following information:

- The name of the failing module, and the module level
- The offset in the module at which the failure occurred

- The instruction at that offset
- The abend type.

This section tells you how to find the information just listed, and contains the following topics:

The documentation you need

The primary documentation you need for investigating abends is the system dump that was 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 if system dumping is not otherwise disabled, a system dump is 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 do not get a dump when an abend occurs" on page 214, and follow the advice given there. When you are sure that dumping is enabled for the appropriate system dump code, you need to re-create the system abend.

You can use the interactive problem control system (IPCS) to process dumps and view them online. See "Formatting system dumps" on page 39 for guidance about processing dumps using IPCS VERBEXIT parameters. 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 contains summary information about the error (search for the eye-catcher ===KE). The internal trace table contains the exception trace entry (if any) that was made at the time the error was detected (search for the eye-catcher ===TR).

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 can format and print the dump offline. For details of how to do this, see Dump utilities (DFHDU680 and DFHPD680).

You might 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 format the same areas in more detail.

Interpreting the evidence

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

Procedure

1. Look at any messages that accompany a CICS system abend, because they might point directly to the cause of the failure.

   For advice about the user response to a message, see CICS messages.
2. Examine the exception trace entry.

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

For details of trace entries, see Using CICS trace.

3. Look at the symptom string in the system dump.

   The symptom string, similar to the short symptom string at the beginning of a transaction dump, appears at the beginning of a CICS system dump. It is also written to SYS1.LOGREC and is issued as part of message DFHME0116.

   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 9 on page 99. 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.

Table 9. 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. Among other things, it might contain the abend code. If you have not already done so, look in CICS messages to see what action it suggests for this abend code.

   If the system cannot 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

After you look at the symptom string at the start of the dump, the next place to look is the kernel domain storage summary.

Procedure

1. Gather the following information from the kernel domain storage areas:
   - A summary of tasks and their status, and whether 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.
2. When you have this information, find out which tasks are associated with the error.

Finding which tasks are associated with the error

To find out which tasks are associated with the error, you can use the kernel task summary. The kernel task summary shows which tasks were in the system when the dump was taken, whether those tasks were running, and whether they were in error.

About this task

The task summary is in the form of a table, where each line in the table represents 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 14 on page 101 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    TCB                CURRENT_PSW

0001   290F5000 KTCB Step    00000000                00000000 29178038        009C3070  078D1000 80000000 00000000 28F71084
0002   290F5680 KTCB QR      00000000                2917CE00 2917B200        009CE9A8  070C0000 80000000 00000000 00FF3B68
0003   29112000 KTCB RO      00000000                2917CF00 2917A168        009CEBD8  078D1000 80000000 00000000 28F189C8
0004   29112680 KTCB CO      00000000                291CBF00 29278F68        009CE778  078D1000 80000000 00000000 28F189C8
0005   2912F000 KTCB FO      00000000                291D7300 291790D0        009FC0F8  078D1000 80000000 00000000 28F189C8
0006   2912F680 Not Running  00000000                2928D080 2917A168
0007   2914C000 Not Running  29392700 00043  CSNE    292E6E00 2917B200
0008   2914C680 KTCB SL      00000000                291D7D00 3BF15000        009ABB88  078D1000 80000000 00000000 28F189C8
0009   29169000 Not Running  00000000                2928D500 2917B200
000A   2931F800 KTCB CQ      00000000                292AC300 29299F68        009ABE88  078D1000 80000000 00000000 29DDD7BA
000B   2A8AF100 Not Running  29394100 00035  CISE    2928D800 2917B200
000C   2A280100 Unused
000E   2A8BC100 Not Running  29393100 00036  CISM    2928DB00 2917B200
000F   2A280800 Unused
0010   2A8BC800 Not Running  29393700 00037  CISP    2928DC80 2917B200
0011   2A87F100 Not Running  2938E100 00007  CSSY    3BF78380 2917B200
0012   292FD000 Not Running  2938D700 00006  CSSY    292E6B00 2917B200
0013   2A29D100 Unused
0014   2A29D800 Unused
0016   2A2BA100 Unused
0017   292FF680 Not Running  2938C700 00004  CSOL    2928D380 3BF15000
0018   3BF5B000 Not Running  2938D100 00005  CEPM    292E6500 3BF74F68
0019   2A2BA800 Unused
001B   2A7FF800 Not Running  00000000                2928DE00 2917B200
001C   2A87F800 Not Running  2938B700 TCP    CSTP    3BF78B00 2917B200
001E   3BFB3000 KTCB SP      00000000                3BF96000 3BF42F68        009AB488  078D1000 80000000 00000000 28F189C8
0020   3BFD0000 KTCB EP000   00000000                292ACE00 3BF74F68        009A9E88  078D1000 80000000 00000000 28F189C8
0021   2A8AF800 Not Running  2938F100 00025  CFQR    3BF78200 2917B200
0023   2A2D7100 Unused
0024   3BFFC680 Not Running  2938C100 00031  CSHQ    3C09D380 2917B200
0026   2A2D7800 Unused
0027   2A2F4100 Unused
0029   3C054000 KTCB L8000   00000000                291D7E00 292BB000        009A3E88  078D1000 80000000 00000000 28F189C8
002B   3C071000 ***Running** 00000000                2928D980 29299F68        009ABE88  078D1000 80000000 00000000 29DDD7BA
002C   3C071680 Not Running  2938E700 00034  CISR    2928D200 2917B200
002E   3C08E680 Not Running  00000000                3C09DE00 2917B200
002F   2A2F4800 Unused
0030   2A2FE100 Unused
0031   2A910100 Not Running  29391700 00027  CSZI    292E6980 3BF88F68
0032   2A2FE800 Unused
0033   2A94B800 KTCB SZ      00000000                3BFD0700 3BF88F68        009A4348  070C0000 80000000 00000000 010B2C90
0036   3C103680 KTCB SO      00000000                291D7F00 3BF17F68        009AB788  078D1400 80000000 00000000 28F189C8
0037   2931F100 Not Running  00000000                3C09D080 2917B200
0038   2A2FF100 Unused
.
.
.
008F   2A7FC800 Unused
0090   2A7FF100 ***Running** 29390700 00049  6421    3BF2F080 2917B200 *YES*  009CE9A8  070C0000 80000000 00000000 00FF3B68
0091   2A8EB100 Not Running  2938F700 00024  CFQS    292E6C80 2917B200
0092   2A8EB800 Not Running  29391100 00026  CSNC    3C09D980 2917B200
0094   2A910800 Not Running  29392100 00023  CEPF    292E6380 3BF01000
0097   2A94B100 KTCB EP001   00000000                3BF79900 3BF01000        009A9390  078D1000 80000000 00000000 28F189C8
```

Figure 14. Kernel task summary showing a task in error

Procedure

1. Locate the task summary table in the formatted dump, and look in the ERROR column. If you find a value of *YES* for a specific task, that task was in error at the time that 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 102). Error data is also captured in the error table even when no system dump is taken at all.

   In Figure 14 on page 101, you can see that kernel task number 0090 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**, meaning that the task was running when the system dump was taken. If more than one task is shown to be running, the different tasks are attached to separate TCBs.

- `Not Running`, meaning that the task is in the system but is currently not running. For example, the task might be suspended because it is waiting for a resource, or the task might be ready to run but waiting for a TCB to become available.
- KE_KTCB, referring to CICS control blocks that correspond to the CICS TCBs. These are treated as tasks in the kernel task summary.
- `Unused`, meaning either that the task was in the system but it has now terminated, or that 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.

It is very likely that the task shown to be in error has a status of "Running", as in the example of Figure 14 on page 101. Such a task would have been running at the time the error was detected.

It is possible, but less likely, that the task shown to be in error has a status of "Not Running". This situation might occur if the task in error was attempting recovery when, for some reason, it was suspended.

3. If you are using trace to help you diagnose a problem, use the TRAN_# and KE_NUM columns of the kernel task summary to find more information about the task.

   The TRAN_# column for a task can contain the following information:

   - 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 such as 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 when that task passes control to CICS.

   To identify the CICS processing associated with the user task, 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 so that you can distinguish the CICS processing associated with the task you are interested in from other CICS processing.

Finding more information about the error

The summary information for the task in error follows the kernel task summary and provides more information about the failure. It provides a storage report for the task, including registers and PSWs, and any data addressed by the registers.

About this task

The PSW is the program status word that is used by the machine hardware to record the address of the current instruction being executed, the addressing mode, and other control information. Figure 15 on page 103 is the first part of an example of such a storage report for a program check.

```
==KE: Tasks in Error: Error Data follows.

** Task in Error: Error Data follows.

=KE: Error Number:  00000001

 KERRD 2A7FF4B0 KERNEL ERROR DATA

    0000  F0C3F461 C1D2C5C1 018400C4 00000000  C4C6C8C1 D7D3C9F1 00000000 2998BA00  *0C4/AKEA.d.D....DFHAPLI1.....q..*
2A7FF4B0
    0020  00000000 29390700 00000000 2A7FF100  3BF2F080 00000001 00000004 FFFFFFFF  *............"1..20............*
2A7FF4D0
    0040  079D1001 80000000 00000000 2A1E7742  00040004 00000000 00000000 00000000  *..............................*
2A7FF4F0
    0060  90800000 00000000 00000000 C6F4E2C1  00000000 2AA00740 00000000 0008196E  *............F4SA....... .......>*
2A7FF510
    0080  00000000 29992B78 00000000 2A8411D0  00000000 00000002 00000000 29390988  *.....r.......d.}..............h*
2A7FF530
    00A0  00000000 2A1E7564 00000000 2AA00690  00000000 000C0000 00000000 2AA00048  *..............................*
2A7FF550
.
.
.
    02A0  40A63D70 0000000E 00010004 C0000000  00000000 2A1E768A 2917B200 0000049C  * w.........{...................*
2A7FF750
    02C0  00000000 00000000 00000000 00000000                                      *................                *
2A7FF770

   Error Code:  0C4/AKEA    Error Type:  PROGRAM_CHECK           Timestamp:  CA434E0E38DF218F

   Date (GMT)   :  03/10/12         Time (GMT)   :  09:52:18.791922
   Date (LOCAL) :  03/10/12         Time (LOCAL) :  10:52:18.791922

   KE_NUM:  0090  KE_TASK:  00000000_2A7FF100  TCA_ADDR:  29390700  DS_TASK:  3BF2F080  XM_TOKEN:  2930A500  TRAN_NO:  00049

=KE: KTCB Details:

   KTCB_ADDR:    2917B200           KTCB_TYPE:    Q                  KTCB_MODE:    QR                 MVS_TCB_ADDR: 009CE9A8
   ACCUM_TIME:   00000000761B5E9B   STIMER_TIME:  000000007D000000   TIMER_STATE:  C0000003           ESTAE_STATE:  00
   ABEND_999:    00

 Error happened in program DFHGA680 at offset 00000322

 Error happened under the CICS RB.
```

Figure 15. Storage report for a task that has experienced a program check - part 1

Procedure

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

   ```
   ==KE: Tasks in Error: Error Data follows.
   ```

2. Next, you will see the kernel error number for the task.

   The kernel assigns error numbers consecutively, starting from 00000001.

   ```
   =KE: Error Number:  00000001
   ```

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

3. Optional: Some kernel error data follows. Usually, you do not need to find the format of this data, but if you do, refer to KERRD - Kernel error data.

4. The next area of interest is the interpretation by the kernel of what went wrong.

   This information includes the error code, the error type, the name of the program that was running, and the offset in the program.

   - The error code shows the system and user completion codes issued when the abend occurred.
   - The error type shows whether the error was associated with, for example, a program check, a system abend, or an internal request for system recovery.

5. There is a report of where the system has recorded that the error occurred, and the circumstances of the failure.

   The information has the following general format:

   ```
   Error happened in program PPPPPPPP at offset xxxxxxxx
   ```

The program name (*PPPPPPPP*) and offset (*xxxxxxxxx*) 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 report shows the following text:

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

The reported program name (*PPPPPPPP*) is the program that owns the current kernel stack entry for the abending task. If this text is shown, it might be possible to locate the failing program using the method described in "Using the linkage stack to identify the failing module" on page 108. The failing program name and offset are also displayed in the section of the report immediately after the contents of the registers are reported. This information has the following format: :

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

If the failing program cannot be located, the module name and offset are reported as unknown. Possible reasons why the program cannot be located are as follows:

- The failure occurred in a z/OS 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.

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

6. After the interpretation by the kernel of the error, one of the following diagnostic messages is shown:
   - `Error happened under the CICS RB`

     The error was detected either when CICS code was executing, or when an access method called by CICS was running (for example, VSAM or QSAM). The CICS RB is the CICS request block, an MVS control block that records the state of the CICS program.
   - `Error did not happen under the CICS RB`

     This message can be issued in any of the following circumstances:
     - An error occurs in CICS SVC code.
     - An error occurs in a CICS z/OS Communications Server exit.
     - CICS detects a runaway task during the execution of an MVS service request.
     - An error occurs during the execution of an SVC request that was made by CICS or an access method invoked by CICS.
7. After either of these messages, data that is likely to be related to the problem is shown. The data shown depends on whether the error happened under the CICS request block.

   - If the error happened under the CICS RB, the error data in the task storage report is based on values in the PSW and the CICS registers at the time the error was detected. Figure 15 on page 103 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 under the CICS RB.
   - If the error did not happen under the CICS RB, for example when CICS was calling an MVS service, you get data based on two sets of registers and PSWs. The registers and PSW of the CICS RB at the time of the error constitute one set. The registers and PSW of the RB in which the error occurred constitute the other set. This data will relate, very probably, to the execution of an SVC routine called by CICS. The error might have occurred, however, during an IRB interrupt or in an SRB. You can confirm whether this has happened by checking flags KERNEL_ERROR_IRB and KERNEL_ERROR_SRB_MODE.

   For more information about this data in the task storage report, see "The storage addressed by the CICS registers and PSW" on page 105.

The storage addressed by the CICS registers and PSW

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

Figure 16 on page 105, Figure 17 on page 106, and Figure 18 on page 107 show parts of an example storage report for a task that failed when a program check was detected.

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 change because, for example, a program check was 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 Breaking Event Address is the address of the last successful branch before the program check. If it is available, you can use it to help determine the location of a wild branch instruction.

```
CICS Registers and PSW

  PSW:  079D1001 80000000 00000000 2A1E7742   Instruction Length:  4  Interrupt Code:  04
  Exception Address:  00000000_00000000

  Execution key at Program Check/Abend: 9     Addressing Mode:  64

  Space at Program Check/Abend: Subspace

  Breaking Event Address: 00000000_2A1E768A - offset 0000026A in module DFHGA680

  64-BIT REGISTERS 0-15

  GPR 0-3   00000000_C6F4E2C1  00000000_2AA00740  00000000_0008196E  00000000_29992B78
  GPR 4-7   00000000_2A8411D0  00000000_00000002  00000000_29390988  00000000_2A1E7564
  GPR 8-B   00000000_2AA00690  00000000_000C0000  00000000_2AA00048  00000000_2AA00100
  GPR C-F   00000000_2A1E77A8  00000000_2AA00690  00000000_2A1E75A2  00000000_00000000

  ACCESS REGISTERS 0-15

  AR  0-3   009FF890  00010004  00000000  00000000
  AR  4-7   00000000  00000000  00000000  00000000
  AR  8-B   00000000  00000000  00000000  00000000
  AR  C-F   00000000  00000000  00000000  00000000

  FLOATING POINT REGISTERS 0-15

  FPR 0-3   00000000_00000000  00000000_00000000  00000000_00000000  00000000_00000000
  FPR 4-7   00000000_00000000  00000000_00000000  00000000_00000000  00000000_00000000
  FPR 8-B   00000000_00000000  00000000_00000000  00000000_00000000  00000000_00000000
  FPR C-F   00000000_00000000  00000000_00000000  00000000_00000000  00000000_00000000
  FPCR      00000000
```

Figure 16. Storage report for a task that has experienced a program check - part 2

The format of the PSW is described in z/Architecture 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 thus its offset within the module, and also the abend type. You find the identity of the failing module itself by examining the kernel linkage stack, as described in "Using the linkage stack to identify the failing module" on page 108.

```
   Data at PSW: 00000000_2A1E7742    Module: DFHGA680   Offset: 00000322

 PSWDATA 2A1E7420

    0000  C4C6C8C7 C1F6F8F0 E3F0021C 001758F0  F0D058F0 F01458F0 F00C58FF 004407FF  *DFHGA680T0.....00}.00..00.......*
2A1E7420
    0020  5CC6C9D3 D3C9D55C A7F40089 23C1D4F6  F4F2F0F1 404DC95D 40F1F061 F0F361F1  **FILLIN*x4.i.AM64201 (I) 10/03/1*
2A1E7440
    0040  F240F1F0 4BF5F040 A9D6E2F6 F8F04040  40C3C9C3 E240F5F6 F5F560E8 F0F4404D  *2 10.50 zOS680   CICS 5655-Y04 (*
2A1E7460
    0060  C35D40C3 D6D7E8D9 C9C7C8E3 40C9C2D4  40C3D6D9 D7D6D9C1 E3C9D6D5 6B406B40  *C) COPYRIGHT IBM CORPORATION, , *
2A1E7480
    0080  C1D3D340 D9C9C7C8 E3E240D9 C5E2C5D9  E5C5C44B 40E44BE2 4B40C7D6 E5C5D9D5  *ALL RIGHTS RESERVED. U.S. GOVERN*
2A1E74A0
    00A0  D4C5D5E3 40E4E2C5 D9E240D9 C5E2E3D9  C9C3E3C5 C440D9C9 C7C8E3E2 406040E4  *MENT USERS RESTRICTED RIGHTS - U*
2A1E74C0
    00C0  E2C56B40 C4E4D7D3 C9C3C1E3 C9D6D540  D6D940C4 C9E2C3D3 D6E2E4D9 C540D9C5  *SE, DUPLICATION OR DISCLOSURE RE*
2A1E74E0
    00E0  E2E3D9C9 C3E3C5C4 40C2E840 C7E2C140  C1C4D740 E2C3C8C5 C4E4D3C5 40C3D6D5  *STRICTED BY GSA ADP SCHEDULE CON*
2A1E7500
    0100  E3D9C1C3 E340E6C9 E3C840C9 C2D440C3  D6D9D74B 40D3C9C3 C5D5E2C5 C440D4C1  *TRACT WITH IBM CORP. LICENSED MA*
2A1E7520
    0120  E3C5D9C9 C1D3E240 6040D7D9 D6D7C5D9  E3E840D6 C640C9C2 D400EBEC D0080024  *TERIALS - PROPERTY OF IBM...}...*
2A1E7540
    0140  A7150007 00000202 2A1E78C8 0000E3F0  10040017 0DEFC0C0 00000119 B90400D1  *x.........H..T0......{{.......J*
2A1E7560
.
.
.
    03E0  00000816 00000817 00000818 00000819  0000081A 0000081B 0000081C 0000081D  *............................*
2A1E7800
    0400  0000081E 0000081F C1D4F3F1 E7F2F0F1  E7D7C3E3 C1404040 C1D4F6F4 F2F0F140  *........AM31X201XPCTA   AM64201 *
2A1E7820
    0420  4410                                                                      *..                             *
2A1E7840

   Data at BEAR: 2A1E768A    Module: DFHGA680   Offset: 0000026A

 BEARDATA 2A1E7420

    0000  C4C6C8C7 C1F6F8F0 E3F0021C 001758F0  F0D058F0 F01458F0 F00C58FF 004407FF  *DFHGA680T0.....00}.00..00.......*
2A1E7420
    0020  5CC6C9D3 D3C9D55C A7F40089 23C1D4F6  F4F2F0F1 404DC95D 40F1F061 F0F361F1  **FILLIN*x4.i.AM64201 (I) 10/03/1*
2A1E7440
    0040  F240F1F0 4BF5F040 A9D6E2F6 F8F04040  40C3C9C3 E240F5F6 F5F560E8 F0F4404D  *2 10.50 zOS680   CICS 5655-Y04 (*
2A1E7460
    0060  C35D40C3 D6D7E8D9 C9C7C8E3 40C9C2D4  40C3D6D9 D7D6D9C1 E3C9D6D5 6B406B40  *C) COPYRIGHT IBM CORPORATION, , *
2A1E7480
    0080  C1D3D340 D9C9C7C8 E3E240D9 C5E2C5D9  E5C5C44B 40E44BE2 4B40C7D6 E5C5D9D5  *ALL RIGHTS RESERVED. U.S. GOVERN*
2A1E74A0
    00A0  D4C5D5E3 40E4E2C5 D9E240D9 C5E2E3D9  C9C3E3C5 C440D9C9 C7C8E3E2 406040E4  *MENT USERS RESTRICTED RIGHTS - U*
2A1E74C0
    00C0  E2C56B40 C4E4D7D3 C9C3C1E3 C9D6D540  D6D940C4 C9E2C3D3 D6E2E4D9 C540D9C5  *SE, DUPLICATION OR DISCLOSURE RE*
2A1E74E0
    00E0  E2E3D9C9 C3E3C5C4 40C2E840 C7E2C140  C1C4D740 E2C3C8C5 C4E4D3C5 40C3D6D5  *STRICTED BY GSA ADP SCHEDULE CON*
2A1E7500
    0100  E3D9C1C3 E340E6C9 E3C840C9 C2D440C3  D6D9D74B 40D3C9C3 C5D5E2C5 C440D4C1  *TRACT WITH IBM CORP. LICENSED MA*
2A1E7520
    0120  E3C5D9C9 C1D3E240 6040D7D9 D6D7C5D9  E3E840D6 C640C9C2 D400EBEC D0080024  *TERIALS - PROPERTY OF IBM...}...*
2A1E7540
    0140  A7150007 00000202 2A1E78C8 0000E3F0  10040017 0DEFC0C0 00000119 B90400D1  *x.........H..T0......{{.......J*
2A1E7560
.
.
.
    0320  20000700 A7150017 002A0000 C1D4F3F1  F2F0F140 D6A58599 A69989A3 85409686  *....x.......AM31201 Overwrite of*
2A1E7740
    0340  40D9C4E2 C1409596 A3408485 A38583A3  85840A23 00004110 D0B041E0 C11650E0  * RDSA not detected......}..\A.&\*
2A1E7760
    0360  10009680 1000E3F0                                                         *..o...T0                       *
2A1E7780
```

Figure 17. Storage report for a task that has experienced a program check - part 3

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. If their values can represent 31-bit addresses, you see the data around those addresses. If their values can represent 64-bit addresses, you see the data around those addresses.

The contents of a register might represent a 24-bit address, a 31-bit address, and a 64-bit address. In that case, you get three sets of addressed data. A lower half of the 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. The 64-bit register could also contain a genuine 64-bit address.

```
   Data at Registers

   REG 0   00000000_C6F4E2C1

 64-bit data cannot be accessed **

 31-bit data cannot be accessed **

 24-bit data follows:

 REGDATA 00F4E2C1

   -0080  00000004 08001000 00000000 03000028  B6A356E0 000000C9 D5E3C5D9 D5C1D300  *.................t.\...INTERNAL.*
00F4E241
   -0060  00000000 00000000 00000000 F4E24000  00000000 00000000 00000000 000000C1  *............4S ................A*
00F4E261
   -0040  E2C3C200 FB028000 FDB30000 00000000  00000000 00000000 00000000 00000000  *SCB.............................*
00F4E281
   -0020  00000000 02000000 0100FF00 0000007F  F18E10C0 80000028 E66A0000 00000000  *................"1..{....W.......*
00F4E2A1

    0000  00000000 4D1138CA 30B230C6 0AB4A400  00000080 9FDA18CA 30B20E00 00000000  *....(......F..u.................*
00F4E2C1
    0020  9FD0C0FF FF010000 00000000 9FDB6000  2600C600 00000000 00000000 9FE04000  *.}{...........-...F.........\ .*
00F4E2E1
    0040  00000000 00000000 00000040 00000002  78760002 78472800 00038000 00000000  *.......... ....................*
00F4E301
    0060  00000000 00000000 00000000 00000000  F9DB4000 00000000 00000000 00000000  *.................9. .............*
00F4E321
    0080  00000700 00000000 00000000 098DA600  00000000 00000000 00000000 00000000  *..............w.................*
00F4E341
    00A0  9FEAF8FF FFFFFF00 00000000 00000000  00000000 00000002 8040807F FCC0007F  *..8..................... ."."{."*
00F4E361
    00C0  FD7B6000 02000100 07000200 9FEB0800  00000000 00000000 00000000 9FF890C0  *.#-...........................8.{*
00F4E381
    00E0  00000002 C2D00000 00000000 000000CA  30B23049 6D1DA300 00000000 00000000  *....B}..............._.t.........*
00F4E3A1

   REG 1   00000000_2AA00740

 64-bit data follows:

 REGDATA 00000000_2AA00740

   -0080  00000000 29992B78 00000000 2A8411D0  00000000 00000002 00000000 29390988  *.....r.......d.}..............h*
2AA006C0
   -0060  00000000 2A1E7564 00000000 2AA00690  00000000 00089FC4 00000000 2AA00048  *.......................D........*
2AA006E0
   -0040  00000000 2AA00100 00000000 2A1E77A8  00000000 2AA00048 00000000 00000000  *...............y................*
2AA00700
   -0020  00000000 2AA00100 00000000 00000000  00000000 2AA00690 00000000 00000000  *................................*
2AA00720

    0000  2A1E787C AAA00840 00000000 00000000  00000000 00000000 00000000 00000000  *...@... ........................*
2AA00740
    0020  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
2AA00760
    0040 -    00FF LINES SAME AS ABOVE
2AA00780

 24-bit data cannot be accessed
```

Figure 18. Storage report for a task that has experienced a program check - part 4

If, for any reason, the register does not address any data, one of the following messages is issued:

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

This means that the addresses cannot be found in the system dump of the CICS region. MVS keeps a record of how CICS uses storage, and any areas not used by CICS are considered to lie outside the CICS address space. Such areas are not dumped in an MVS SDUMP of the region.

It is also possible that the addresses were in the CICS region, but were not included in the SDUMP. This is because MVS enables you to take SDUMPs selectively, for example "without LPA". If a selective SDUMP occurred without your knowledge, you might think you had an addressing error when the address was actually valid.

Using the linkage stack to identify the failing module

A linkage stack for a task represents the sequence in which modules and subroutines were 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.

About this task

After you find which task was in error from the kernel's task summary (see "Finding which tasks are associated with the error" on page 100), you need to find out which module was in error. If you report a problem to the IBM Support Center, the information you need to provide includes the module name.

Figure 19 on page 108 shows a typical kernel linkage stack.

```
KE_NUM @STACK   LEN  TYPE ADDRESS  LINK REG OFFSET ERR NAME

0090   2A83F040 01E0 Bot  28F03C00 A8F04230 000630     DFHKETA
0090   2A83F220 03E0 Dom  28F20900 A8F20B76 000276     DFHDSKE
0090   2A83F600 1130 Dom  28F54BC8 A8F56168 0015A0     DFHXMTA
0090   2A840730 0AA0 Dom  296150C8 A9616546 00147E     DFHPGPG
                     Int   +0003CA A96152AE 0001E6     INITIAL_LINK
0090   2A8411D0 0E90 Dom  2998BA00 A9380644 000000 *Y* DFHAPLI1
                     Int   +002B4E A998C77A 000D7A     CICS_INTERFACE
0090   2A842060 06C0 Sub  29380A58 A9381E2E 0013D6     DFHSRP
0090   2A842720 1290 Dom  28FAB880 28FB02CD 004A4D     DFHMEME
                     Int   +003D82 28FABB6A 0002EA     SEND
                     Int   +001840 28FAF6EA 003E6A     CONTINUE_SEND
0090   2A8439B0 06F0 Dom  2906B5B8 A906D0E0 001B28     DFHDUDU
                     Int   +000C6C A906B7CA 000212     SYSTEM_DUMP
                     Int   +001AE6 A906C63E 001086     TAKE_SYSTEM_DUMP
```

Figure 19. Example of a kernel linkage stack showing a task in error

Procedure

1. 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.
2. When you have found the task number, look at the TYPE column.

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

   - `Bot` marks the first entry in the stack.
   - `Dom` marks a stack entry caused by a domain call.
   - `Sub` marks a stack entry caused by a subroutine.
   - `Lifo` marks a stack entry caused by a LIFO module.
   - `Int` marks a call to an internal procedure identified to the kernel.
3. 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 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 in Figure 19 on page 108, program DFHAPLI1 is shown to be in error.
4. If module DFHAPLI or DFHAPLI1 is flagged as in error, consider first whether an application is the cause of 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.
5. If an application is not the cause of the error, or the module flagged in error is not DFHAPLI or DFHAPLI1, report this module name 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 109.

What to do next

You can sometimes use the technique described in this section to gather the information that the IBM Support Center needs to resolve a 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 valid only 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 the following 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.

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.

Before you begin

For details about format of the PSW, or how to calculate the offset, see z/Architecture Principles of Operation.

About this task

If you report a problem to the IBM Support Center, the information you need to provide includes the instruction at the offset.

Procedure

1. 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.
2. Identify the abend type from the program interrupt code, so that you can report that too.

   For example, it might be protection exception (interrupt code 0004), or data exception (interrupt code 0007). For a list of program interrupt codes (PICs), see “What type of program check occurred?” on page 90.

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 20 on page 110 shows some entries from a typical program storage map summary.

```
==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
ADDRESS

 DFHCSA   80041800 DFHCSAOF 00041000 680 I0709192 I  07/09 20.52    1       1       CDSA  RPL RESIDENT  24    24
00000048_40A01D88
                   DFHCSA   00041600 680 I0709192 I  07/09 20.52                                          24    24
 DFHTCP   80041C18 DFHTCP   00041B00 680 I2009070 I  20/09 07.50    1       1       CDSA  RPL RESIDENT  24    31
00000048_40A046C8
                   DFHTCORS 00041EB0 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCCOM 00042188 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCCSS 00042550 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCTI  000426C8 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCSAM 00042750 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCAM  00042B60 680 I2009070 I  20/09 07.50                                          24    24
                   DFHTCTRN 00043A50 680 I2009070 I  20/09 07.50                                          24    24
 DFHTCTDY 00044720 DFHTCTDY 00044700 680 I0709192 I  07/09 20.14    1       1       CDSA  RPL RESIDENT  24    24
00000048_40A04848
 DFHDUIO  80080000 DFHDUIO  00080000 680 I2009070 09/20/12 07.31    1       1       RDSA  ANY REUSABLE  24    31
00000048_40A01A88
                   IPRDUIO  00081688
 DFHAIP   00082118 DFHEIP   00082000 680 I0210191 I  02/10 20.04    1       2       RDSA  ANY RESIDENT  24    24
00000048_40A01C08
                   DFHEIPA  00086B70 680 I2009070 I  20/09 07.57                                          24    24
                   DFHEIG   00087138 680 I0210191 I  02/10 19.47                                          24    24
                   DFHEIGA  00089AC8 680 I2009070 I  20/09 07.58                                          24    24
                   DFHCPI   0008A008 680 I0210191 I  02/10 19.43                                          24    24
                   DFHAICBP 0008AF60 680 I0709192 I  07/09 20.22                                          24    24
 DFHLIRET 8008B314 DFHLIRET 0008B200 680 I0709192 I  07/09 21.15    1       1       RDSA  ANY RESIDENT  24    24
00000048_40A037C8
 DFHDLI   8008B514 DFHDLI   0008B400 680 I2009070 I  20/09 07.28    1       1       RDSA  ANY RESIDENT  24    ANY
00000048_40A05448
.
.
.
```

Figure 20. 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 DEFINE_PROGRAM call for that program. If a REQUIRED_RMODE or REQUIRED_AMODE is not specified, a dsah symbol (–) is shown in the appropriate column. If AMODE_ANY or RMODE_ANY is specified, ANY is shown in the appropriate column. Other values are shown as specified.

Dealing with waits

This section gives you information about what to do if you are aware that a task is in a wait state, or if CICS has stalled.

It contains the following topics:

If CICS has stalled, turn directly to "What to do if CICS has stalled" on page 162.

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 Chapter 2, "Classifying the problem," on page 5.

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 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 section 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 "Dealing with performance problems" on page 204.

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 section 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 195.

Techniques for investigating waits

You can investigate waits in a CICS system by online inquiry, by tracing, or by analysis of the formatted CICS system dump. The last two techniques are, to some extent, complementary.

Online inquiry 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 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.

A CICS system dump can give 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. However, 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 - online method

Online, you can use either CEMT INQ 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 from a user 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 or the resource name identifying the unavailable resource. **CEMT INQ TASK** displays:

- the resource type of the unavailable resource in the HTYPE field.

- the resource name of the unavailable resource in the HVALUE field.

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 16 on page 166 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 **CEMT PERFORM SNAP** or **CEMT PERFORM DUMP** - 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 Using CICS trace.

Setting up trace for wait problems

About this task

Gate DSSR 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 165.) The level-1 trace points **DS 0004** and **DS 0005** are produced on entry to, and exit from, the gate.

Procedure

1. Select tracing to capture the DS level-1 trace entries to investigate a 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.
2. Ensure that tracing is performed for the task that has the wait problem. Next, you can:
   a) Select special tracing for just that task, and disable tracing for all other tasks by setting the main system trace flag off.
   b) Select special tracing for other tasks as well if it becomes clear that they are implicated in the wait.

Interpreting trace for wait problems

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 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 - the formatted CICS system dump

If you are suitably authorized, you can request a CICS system dump using **CEMT PERFORM DUMP**, **CEMT PERFORM SNAP**, or **EXEC CICS PERFORM DUMP**. 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.

The dispatcher task summary gives you information like that shown in Figure 21 on page 113.

```
===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            MVS=WAIT_MVS  SUSP=SUSPEND  OLDC=WAIT_OLDC  OLDW=WAIT_OLDW
     DTA= DISPATCHER TASK AREA
     AD = ATTACHING DOMAIN
     MO = TASK MODE               CO=CONCURRENT  QR=QUASI-REENTRANT  RO=RESOURCE OWNING  SZ=FEPI OWNING
     RP=ONC/RPC OWNING  FO=FILE 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
00000001 06A23900 SY SUS N OK -  ENF      NOTIFY   MVS  17:50:10.056      -       06B69080 DM 06C33640 RO 06C33658
00020003 06A23580 SY SUS N OK -                     SUSP 17:51:23.515      -       06B69180 XM 06C06690 QR 06B69180 06C066900000022C
00040003 06A23200 SY SUS N OK -  TIEXPIRY DS_NUDGE SUSP 17:50:35.325      -       06B69280 TI 003D0003 QR 06B6A530
00820003 06A30900 SY SUS N OK -  ICEXPIRY DFHAPTIX SUSP 17:50:35.326      -       06B89180 XM 06C06360 QR 06B89180 06C063600000006C
00900003 06A30C80 SY SUS N OK -  ICMIDNTE DFHAPTIM SUSP 17:50:29.437      -       06B89880 XM 06C06250 QR 06B89880 06C062500000005C
00940003 06B5FC80 SY SUS N OK -  TCP_NORM DFHZDSP  OLDW 17:51:46.369      -       06B89A80 XM 06C06470 QR 0004FC10 06C064700000008C
00980001 06C3D080 SY SUS N OK IN SMSYSTEM          SUSP 17:50:10.081 17:55:10.081 06B89C80 SM 00000002 QR 06B6A580
0200000B 07128B00 NS SUS Y OK -  ZCIOWAIT DFHZARQ1 SUSP 17:51:39.616      -       06B9C080 XM 06C06580 QR 06B9C080 06C065800000029C
02020009 07135080 NS RUN                                                         06B9C180 XM 06C06140 QR          06C061400000031C
```

Figure 21. 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 the following table.

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 10. 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.

Table 10. Descriptions of fields shown in the dispatcher task summary (continued)	
Field	**Description**
MO	The dispatching mode for the task. There is an MVS TCB for each mode. All tasks in a given mode are running, or will run next, under the TCB corresponding to the mode. The possible values are: CO—concurrent. FO—file owning. QR—quasi-reentrant. RO—resource owning. RP—ONC RPC SZ—FEPI owning.
P	Whether at the time of the suspend call the control block containing the definition of the transaction specified SPURGE(YES) or SPURGE(NO). SPURGE(NO) inhibits deadlock timeout, 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.
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_MVS, WAIT_OLDC, or WAIT_OLDW of gate DSSR. For an explanation of these functions, see "How tasks are made to wait" on page 165. • 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.)

Table 10. Descriptions of fields shown in the dispatcher task summary (continued)	
Field	**Description**
ST *(suspended tasks only)*	The type of function that was invoked to suspend a currently suspended task. Possible values include: MVS—function WAIT_MVS 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 165.
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_MVS 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.
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 timeout type for the task. The possible values, where one is given, are: • IN—a timeout interval has been specified for the task. • DD—deadlock action is to be delayed when the timeout interval expires. • DI—deadlock action is immediate when the timeout interval expires.
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 112 .

Table 11 on page 116 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 11. 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
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_MVS WAIT_OLDC WAIT_OLDW	IN IN IN IN	DSSR
PS	*none*			

Table 11. Parameters and functions that set fields shown in the dispatcher task summary (continued)

Field	Related parameter	Function	Input or output	Gate
RESOURCE NAME	RESOURCE_NAME	ADD_SUSPEND SUSPEND WAIT_MVS 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_MVS 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_MVS WAIT_OLDC WAIT_OLDW	IN IN IN	DSSR
TIME OF SUSPEND	*none*			
TASKNO	*none*			
TIMEOUT DUE (see note 4)	*none*			
TT	INTERVAL *and* DEADLOCK_ACTION	SUSPEND WAIT_MVS 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*			

Note:

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 “Dispatcher task summary fields” on page 113.

2. Parameters ECB_ADDRESS and ECB_LIST_ADDRESS only relate to SUSPAREA when the task has been suspended by the WAIT_MVS, WAIT_OLDW, or WAIT_OLDC functions of gate DSSR.
3. Parameter ECB_LIST_ADDRESS is only valid for functions WAIT_MVS 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.

Investigating terminal waits

Before you begin

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.

About this task

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.

Procedure

1. If you have one or more unresponsive terminals:
   a) Use **CEMT INQ TERMINAL** to find the transaction running at the terminal.
   b) Use**CEMT INQ TASK** to find out what resource that task is waiting on.
   c) When you know that, look at Table 16 on page 166 to find where you can get further guidance.
2. If all the terminals in the network are affected, and CICS has stalled, read “What to do if CICS has stalled” on page 162 for advice about how to investigate the problem.

Results

If you have 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, or a batch region, or it could be another CICS region connected by an interregion communication link, or it could be a system that is connected by an LUTYPE6.1 or APPC (LUTYPE6.2) protocol. If LUTYPE6.1 is in use, the other system might be another CICS region or an IMS region. With APPC (LUTYPE6.2), 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 or a DISOSS system at the other end of the link.

If you eventually find that the fault lies with a terminal, or a resource such as DISOSS, the way in which you approach the problem depends on 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

When you investigate a terminal wait, there are preliminary considerations that might lead to a simple solution.

- Look for an obvious physical explanation for the wait, for example, a terminal operator has not responded to a request for input. For a printer, is it powered off, or has it run out of paper?

- Check in the CSTL and CSNE logs for any messages. If either DFHTCP or DFHZCP detected an error related to terminal control, a message is sent to the CSNE log, and might also be sent to the console.

  If there is a message that reports a terminal error that can be related to the task, the message should provide more information about why the task is waiting. For an explanation of the message, and a description of the system action in response to the error, use the CMAC transaction or see CICS messages.

  The CSNE log entry might show that an error was detected, but that TACP or NACP took no action. This situation might occur if the line or terminal is out of service, or if the error actions are turned off in the user exits of DFHTEP and DFHNEP. In this situation, the CICS code to resume the waiting task would never run, and the task would wait indefinitely.
- Check whether any HANDLE CONDITION routines for terminal errors are coded incorrectly. If there is an attempt to access the terminal with such an error in a routine, the application might wait indefinitely.
- If the terminal is installed using autoinstall, check whether the system loaded DFHZATA, the autoinstall program, or DFHZCQ, which is called by DFHZATA. The system might fail to load DFHZATA or DFHZCQ because of a short-on-storage condition. If so, deal with the cause of the short-on-storage condition.

  Check the delete delay that you have specified; if the delay is too short, your system might be deleting and reinstalling terminals unnecessarily.

  If storage fragmentation prevents DFHZATA or DFHZCQ from being loaded, consider defining them as resident. However, be aware that DFHZCQ is a large program, and check your storage requirements before making this change.

If none of the preliminary considerations apply, start a systematic investigation into the reason for the wait.

Terminal waits - a systematic approach

You first need to 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.

Your strategy must then be to find where in the communication process the fault lies. These are the basic questions that must be answered:

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. **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 either online, using a system programming command, or offline, by looking at the appropriate TCTTE in the formatted system dump.

Online method

Use the transaction CECI to execute the system programming command **EXEC CICS INQUIRE TERMINAL DEVICE**. This returns one of the terminal types that are described in TERMINAL resources.

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 Data areas.

What type of access method is in use?

You can use both an online method and an offline method for finding the type of access method being used by the terminal that is not responding.

Online method

Use the CECI transaction to execute the system programming command EXEC CICS INQUIRE TERMINAL ACCESSMETHOD. 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 Data areas relates values to access methods.

The most common access method is the z/OS Communications Server, identified by a value of TCTEVTAM. The problem determination procedures outlined below focus exclusively on the Communications Server. You might also find either of these values, each being of special significance for problem determination:

- TCTEISMM, meaning that the access method is ISMM. This is used for interregion communication, and it shows that the resource that is not responding is a remote CICS region. In such a case, 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 a physical terminal" on page 127.
- TCTELU6, meaning that you have intersystem communication (ISC). In this case, the resource that is not responding is a remote system, and the way you deal with the wait depends on what the remote system is. If it is a CICS system, you need to 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 read a diagnosis book from another library for advice on problem determination.

If you have any other access method, for example BSAM, you need to adapt the guidance given here accordingly.

z/OS Communications Server in use - debugging procedures

The following information gives guidance about debugging terminal waits when the access method is z/OS Communications Server.

1. Look in the CSNE log to see if there is an error message that explains the wait. If it contains an error code, refer to z/OS Communications Server: SNA Messages.
2. Look 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. For an explanation of SNA sense codes, see z/OS Communications Server: IP and SNA Codes.

Is the problem associated with the z/OS Communications Server?

You can find the z/OS Communications Server (SNA) process status with respect to the waiting task from fields TCTEICIP and TCTEIDIP in the TCTTE.

The following are the values you might find there, and their interpretations:

```
     TCTECIP          command request in progress
     TCTEDIP          data request in progress
```

Either of these status values indicates that a Communications Server request is in progress, and that the communications server RPL is active. A response is expected either from the Communications Server, 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 Communications Server request is not in progress, the next field of interest is in the Communications Server system area of the TCTTE. Find four bytes of Communications Server exit IDs, starting at field TCTEEIDA. If any are nonzero, the Communications Server request has completed. Nonzero values suggest that the Communications Server is not involved in the wait. You can find the meanings of the values from the Communications Server module ID codes list in the table below.

If you suspect that the problem is associated with Communications Server, consider using either CICS Communications Server exit tracing or the Communications Server buffer tracing. Both of these techniques can give you detailed information about the execution of Communications Server requests. For guidance about using the techniques, read the appropriate sections in Using CICS trace.

z/OS Communications Server submodule identifiers

This table contains Product-sensitive Programming Interface information.

Hex ID	Module	Description
X'00'	ZDSP	DISPATCH
X'01'	ZARQ	READ /WRITE R
X'02'	ZLOC	LOCATE
X'03'	ZDET	DETACH
X'04'	ZTCP	TCP
X'06'	ZCRQ	COMMAND REQS
X'08'	ZSTU	STATUS CHANGE
X'09'	ZTSP	TERMINAL SHARING
X'0A'	ZHPX	HPO RPL EXEC OS ONLY
X'0B'	ZISP	ALLOCATE/FREE
X'0C'	ZIS1	INTER SYSTEM
X'0D'	ZIS2	INTER SYSTEM 2
X'0E'	ZABD	INVALID REQUEST/ABEND
X'10'	ZATI	ATI
X'11'	ZATT	ATTACH TASK
X'12'	ZFRE	FREE STORAGE
X'13'	ZGET	GET STORAGE
X'14'	ZRAC	RECEIVE ANY
X'15'	ZRST	RESETSR
X'16'	ZRVS	RECEIVE SPEC
X'17'	ZRVX	RECEIVE S EXT
X'18'	ZSDS	SEND NORMAL
X'19'	ZSDX	SEND DATA EXIT
X'1A'	ZUCT	TRANSLATION
X'1B'	ZUIX	USER EXIT
X'1C'	ZACT	ACTIVATE SCAN
X'1D'	ZSDR	SEND RESPONSE
X'1E'	ZHPS	HPO SEND/RECV CALL

Hex ID	Module	Description
X'1F'	ZRPL	RECV.ANY BLDER
X'20'	ZAIT	ATTACH INIT
X'21'	ZASX	ASYN COM EXIT
X'22'	ZCLS	CLOSE DESTIN
X'23'	ZCLX	CLOSE DS EXIT
X'24'	ZDWE	DWE PROCESS
X'25'	ZLEX	LERAD EXIT
X'26'	ZLGX	LOGON EXIT
X'27'	ZLRP	LOGICAL REC
X'28'	ZLTX	LOSTERM EXIT
X'29'	ZOPN	OPEN DESTINAT
X'2A'	ZOPX	OPEN DESTEXIT
X'2B'	ZRAQ	READAHEAD QUE
X'2C'	ZRAR	READAHEAD RET
X'2E'	ZRRX	REL REQUEST EX
X'2F'	ZNSP	NETWORK SPEC EXIT
X'30'	ZRSY	RESYNC
X'31'	ZSAX	SEND COMM EXT
X'32'	ZSCX	SCIP EXIT
X'33'	ZSDA	SEND ASYN COM
X'34'	ZSKR	SEND COMMAND RESPONSE ID
X'35'	ZSES	SESSIONC COM
X'36'	ZSEX	SESSIONC EXIT
X'37'	ZSIM	SIMLOGON
X'38'	ZSIX	SIMLOGON EXIT
X'39'	ZSLS	SETLOGON START
X'3A'	ZSSX	SEND COM EXIT
X'3B'	ZSYX	SYNAD EXIT
X'3C'	ZTAX	TURNAROUND EXIT
X'3D'	ZTPX	TPEND EXIT
X'3E'	ZOPA	SNA OPEN ACB
X'3F'	ZSHU	SNA SHUTDOWN
X'40'	ZQUE	TERMINAL SHARING
X'41'	ZEMW	ERROR MESSAGE WRITER
X'42'	ZSYN	SYNCPOINT HANDLER
X'43'	ZTRA	SNA RPL TRACE
X'44'	ZAND	ABEND CONTROL BLOCK

Hex ID	Module	Description
X'45'	ZCNA	CONSOLE CONTROL
X'46'	ZCNR	CONSOLE REQUEST
X'47'	ZCNC	CONSOLE ABNORMAL COND.
X'48'	ZUAX	ATTACH USER EXIT
X'49'	ZUOX	OUTPUT USER EXIT
X'4A'	ZARL	LU6.2 APPL REQUEST
X'4B'	ZARM	LU6.2 MIGRATION
X'4C'	ZRVL	LU6.2 RECEIVE
X'4D'	ZRLX	LU6.2 RECEIVE EXIT
X'4E'	ZSDL	LU6.2 SEND
X'4F'	ZSLX	LU6.2 SEND EXIT
X'50'	ZERH	LU6.2 APPL ERP
X'52'	ZBKT	LU6.2 BRACKET STATE M/C
X'53'	ZCNT	LU6.2 CONTENTION STATE
X'54'	ZCHS	LU6.2 CHAIN SEND
X'55'	ZCHR	LU6.2 CHAIN RECEIVE
X'56'	ZUSR	LU6.2 CONVERSATION STATE
X'57'	ZDST	SNA-ASCII TRAN ROUTINE
X'58'	ZEV1	ENCRYPTION VALIDATION 1
X'59'	ZEV2	ENCRYPTION VALIDATION 2
X'5E'	ZXRC	XRF TERMINAL RECOVERY
X'5F'	ZXTS	XRF TERMINAL SCAN
X'60'	ZXRL	LU6.2 Transaction Routing
X'61'	ZINT	Initialization Module Ident
X'62'	ZXRT	LU6.2 Transaction Routing TOS
X'63'	ZSTA	LU6.2 Application Status
X'64'	ZRLP	LU6.2 RECEIVE post-z/OS Communications Server
X'65'	ZCRT	LU6.2 RPL_B state
X'66'	ZRAS	LU6.2 Slow-down processing
X'67'	ZXPS	LU6.2 Per sess recovery
X'7D'	ZRLG	RESPONSE LOGGER
X'7E'	ZNAC	NACP
X'7F'	ZRSP	RESYNC SYSTEM TASK
X'80'	ZATR	ZATR restart deletes
X'82'	ZATA	ZATA autoinstall
X'84'	ZATD	ZATD autoinstall delete

Hex ID	Module	Description
X'86'	ZGMM	GOOD MORNING TRANSACTION
X'8B'	ZATS	ZATS remote install entry
X'C0'	ZQ00	DFHZCQ REQUEST ROUTER
X'C1'	ZQIN	ZC INITIALIZE
X'C2'	ZQBA	ZC Bind Analysis
X'C3'	ZQCH	ZC CHANGE
X'C4'	ZQDL	ZC DELETE
X'C5'	ZQIT	ZC INSTALL TCTTE
X'C6'	ZQRC	ZC RECOVER
X'C7'	ZQRS	ZC RESTORE
X'C8'	ZQIQ	ZC INQUIRE
X'C9'	ZQIS	ZC INSTALL
X'C4'	ZTCT	DUMMY TCTTE IDENTIFIER

z/OS Communications Server in use - is SNA LU control at fault?

The first place to look is 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 INQ 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 either of these values:

```
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.

The z/OS Communications Server in use—is the LU 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. You need now to look at some fields in the TCTTE for the terminal to find its status.

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

Information about the status and attributes of the SNA LUs 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 22 on page 125 shows the index followed by a terminal entry with its interpreted status and attribute information.

```
===TCP: TERMINAL CONTROL SUMMARY (SNA LUs)
        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 22. 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 Data areas to find out what they mean.

The following are the questions that need to be asked, and some values that could provide the answers.

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. Some of the values you might see are:

   ```
   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 TYPETERM definition. This means that EXEC 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 TYPETERM 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. You can tell whether the task has been started from a terminal, or by ATI, from field TCTEICCV:

     ```
     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 in fields TCTTEIC and TCTECTI. The values you might find there are:

```
TCTTEOIC = 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? You can find the task session state with SNA from field TCTEMOST, and, 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:

   ```
   TCTEBPE = 1      bracket protocol is required
   ```

   If you find that bracket protocol is required, look in field TCTEIINB:

   ```
   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      z/OS Communications Server 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.

   You next need to 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? Field TCTEIPRA tells you this:

   ```
   TCTEPRA = 1      the terminal should respond
   ```

If the values you have found 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 system programming command EXEC CICS INQUIRE TERMINAL DEVICE.

Tools you can use for debugging terminal waits when the z/OS Communications Server is in use

Among your debugging tools, two are likely to be of particular use for investigating terminal waits in an SNA environment.

These tools are as follows:

- Communications Server SNA buffer trace. This trace feature is part of the Communications Server itself and you must see the appropriate manual in the Communications Server library for details of how to use it.

- CICS Communications Server exit trace. This trace feature is part of CICS and you can control it from the CETR panel.

For a description of the use of these two types of tracing in CICS problem determination, see Using CICS trace.

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, you need to 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.

If a session has been acquired and it 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 region is not responding because the related task there has been suspended.

You need to identify the related task in the remote region, and find out the resource it is waiting on. When you have done that, see Table 16 on page 166 to find out which part of this section to turn to next.

Investigating the related task in a remote region

You can identify the region that is not responding to your local task, the related task in the remote region that is not responding, and the resource that the remote task is waiting on.

Procedure

1. 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 INQ TASK. The first four characters give you the SYSIDNT of the remote CICS region.
   - **If the task is using intersystem communication** (ISC), you need to 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 region.
2. When you have identified the region, take a system dump of it.

   You can do that either by using the **CEMT PERFORM DUMP** command in that region, or by using the MVS DUMP command.
3. Take a system dump of the local region.
4. Format the dumps using the formatting keyword DS to get a summary of the tasks in each region, and TCP to get the TCTTEs.
5. Find the TCTTE for the task in the local region.

   The way you find the TCTTE for the task in the remote 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 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 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.

6. 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.

7. 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.

8. When you have identified the resource, turn to the appropriate section in this section for guidance about investigating waits on that resource.

Investigating storage waits

If a task is waiting for a long time on any of the resource types CDSA, RDSA, SDSA, UDSA, ECDSA, ERDSA, ESDSA, ETDSA, EUDSA, GCDSA, GUDSA, or GSDSA, you can use a CICS system dump to investigate further.

About this task

Waits on these resources occur when tasks make unconditional storage requests (SUSPEND=YES) that cannot be satisfied. For storage requests below the 16 MB line, waits can be on the CDSA, RDSA, SDSA, or UDSA. For storage requests above 16 MB but below 2 GB, waits can be on the ECDSA, ERDSA, ESDSA, ETDSA, or EUDSA. For storage requests above the bar, waits can be on the GCDSA, GUDSA, or GSDSA.

If tasks make conditional storage requests (SUSPEND=NO), tasks are not suspended on these resources. If the request cannot be satisfied, an exception response is returned.

CICS automatically attempts to relieve storage when it is under stress, for example by releasing storage occupied by programs whose current use count is 0. Also, if a task waits for storage longer than the deadlock timeout parameter specified in the installed transaction definition, the task might be automatically purged. Certain conditions prevent purging of a task, for example, a deadlock timeout value of 0, or a specification of SPURGE(NO).

The most likely reasons for extended waits on storage requests are as follows:

- The task issued an unconditional GETMAIN request for an unreasonably large amount of storage.
- The task issued an unconditional GETMAIN request for a reasonable amount of storage, but the system has too little available. The system might be approaching a short-on-storage (SOS) condition, or the storage might be too fragmented to satisfy the request.

Procedure

1. Produce a CICS system dump, and format it using the formatting keyword SM.
2. Interpret the dump to investigate the problem.
   a) Was the request for too much storage? Look in the SM suspend queue summary.

      This summary includes the number of bytes requested by every task that the storage manager has suspended. You can see whether any task 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 made a reasonable GETMAIN request, investigate whether the system is approaching an SOS condition.

b) Is the storage close to being exhausted? Look at the DSA summary in the formatted dump.

This summary shows the current free space in each DSA, both in bytes and as a percentage of the total storage. It also shows the size of the largest free area, that is, the biggest piece of contiguous storage. (In this context, contiguous storage means storage that is 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 probably 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 might indicate that the task 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, see "Dealing with loops" on page 196 for more information.

If the task made a reasonable storage request and the system seems to have sufficient free storage, investigate whether fragmentation of free storage is causing the GETMAIN request to fail.

c) Is fragmentation of free storage causing the GETMAIN request to fail? Look at the DSA summary in the formatted dump.

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

Investigating temporary storage waits

If a user task is waiting on a resource type that starts with TS, showing that it is for temporary storage, the following information describes possible reasons for the wait.

About this task

Enqueues on temporary storage should be held in retained state, so no user task should ever wait on a resource type of ENQUEUE with a value of TSNQ (temporary storage enqueue). If you have this resource type, see "Investigating enqueue waits" on page 131.

Procedure

- **Resource type TSAUX** indicates that a task is waiting because it has made an unconditional request for temporary storage, and the request cannot be met because insufficient auxiliary storage is available. Take the following actions:

  a) Get a system dump and format it using the formatting keyword TS to show the temporary storage control blocks, as well as SM and KE as the information for these components could be useful.

  b) Analyze the dump using the help given in "Is temporary storage close to being exhausted?" on page 131 and "Is fragmentation of unallocated storage causing the WRITEQ TS request to fail?" on page 131.

  These are the two most likely reasons why a task that has issued an unconditional 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 task has issued an **`EXEC CICS WRITEQ TS`** command, 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 deadlock timeout 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 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.

- **Resource type TSBUFFER** indicates that the task that is waiting 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** 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, check if there is a hardware fault or a problem with VSAM.
- **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, check if there is a hardware fault or a problem with VSAM.
- **Resource type TSMAINLM** indicates that the task that is waiting issued a main temporary storage request, but the request cannot be met because insufficient storage is available. If you find that tasks are often made to wait on this resource, check for large temporary storage queues that could be deleted, or consider increasing the limit for the storage that is available for main temporary storage queues to use (system initialization parameter **TSMAINLIMIT**).
- **Resource type TSPOOL** indicates that the maximum number of concurrent requests (10) for a temporary storage pool in the coupling facility has been reached. The task resumes when one of the requests completes.
- **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 has the lock on the queue. The length of time that a task has the lock on a temporary storage queue depends on whether or not the queue is recoverable. 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 UOWs, or making conversational tasks pseudoconversational.
- **Resource type TSSHARED** indicates that a shared temporary storage request is being processed asynchronously. The task is resumed when the request is complete. The logic that determines whether a request is processed synchronously or asynchronously is outside CICS control.
- **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 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.

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 trace. 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 "Dealing with loops" on page 196.

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?

You can tell whether fragmentation of unallocated temporary storage is causing the WRITEQ TS request to fail by looking at the temporary storage summary in the dump.

The following fields in the summary are of interest should your task be suspended on resource type TSAUX:

```
Number of control intervals in data set:
Number of control intervals currently in use:
Available bytes per CI:
```

For control intervals of 4K, the `available bytes per CI` figure is 4032.

If your task is attempting to a write a record that is smaller than or equal to the `available bytes per CI` figure (including its record header which is 28 bytes long), this means that no control interval has the required amount of contiguous space to satisfy the request.

If your task is attempting to write a record that is longer than the `available bytes per CI` figure, CICS splits the record into sections of a length equal to this figure. CICS then attempts to store each section in a completely empty control interval, and any remaining part of the record in a control interval with the contiguous space to accommodate it. If your task is waiting on resource type TSAUX after having attempted to write a record longer than the `available bytes per CI` figure, either of the following has occurred:

- There are not enough available completely empty control intervals to accommodate all the sections

  ```
  (CIs in data set - CIs in use) < (record length / available bytes per CI)
  ```

- No control interval has enough contiguous space to accommodate the remainder.

Investigating enqueue waits

A task is suspended by the enqueue domain if it requests access to a resource on which another task already holds an enqueue (lock).

About this task

There are two ways in which you can discover the owner of the enqueue that the task is waiting on:

Procedure

- Use the **`CEMT INQUIRE UOWENQ`** command.
  For an example of how to use this command to discover the owner of an enqueue, see "Resource type ENQUEUE" on page 193.

For definitive information about **CEMT INQUIRE UOWENQ**, see CEMT INQUIRE UOWENQ in the IBM Knowledge Center.

Do not use the **EXEC CICS ENQ** command for recoverable resources.

- Use the NQ section of a system dump if you have an enqueue wait for one of the following resource names:
  - ISSSENQP
  - JOURNALS
  - KCADDR
  - KCSTRING
  - LOGSTRMS

  Some types of enqueue wait are not displayed by the CEMT INQUIRE UOWENQ command. In these cases, you can use a system dump to identify the owner of the enqueue (for an example, see "Using a system dump to resolve enqueue waits" on page 132) .

Results

It is possible for an enqueue wait to be caused by a deadlock. For more information on how to resolve a deadlock, see "Resolving deadlocks in a CICS region" on page 158.

Using a system dump to resolve enqueue waits

The **CEMT INQUIRE UOWENQ** (or **CEMT INQUIRE ENQ**) command does not return information about enqueues on some types of resources.

Table 12 on page 132 shows the resources that this applies to.

Table 12. Resources for which INQUIRE UOWENQ does not return information

Resource name	Type of resource
ISSSENQP	Sockets used by IPIC communication for an IPCONN.
JOURNALS	CICS journal names used during creation, deletion, or use of a journal entry. See "Log manager waits" on page 187 for information to help you diagnose problems with the MVS system logger.
KCADDR	Addresses locked internally by CICS.
KCSTRING	Strings locked internally by CICS.
LOGSTRMS	MVS logstream names used during connection of streams to the MVS logger. A long wait could indicate a problem with the logger. See "Log manager waits" on page 187 for information to help you diagnose problems with the MVS system logger.

To investigate enqueue waits on these resources, you can use the NQ section of a system dump. (You can use a system dump to investigate enqueue waits on other types of resource, but you might find the INQUIRE UOWENQ command more convenient.)

CICS maintains a separate enqueue pool for each type of resource that can be enqueued upon. To produce a summary of each enqueue pool, specify 1 on the NQ dump formatting keyword (dump formatting keywords are described in "Summary of system dump formatting keywords and levels" on page 40). Figure 23 on page 133 shows an example summary for the transient data enqueue (TDNQ) pool.

```
==NQ: ENQUEUE POOL SUMMARY - TDNQ

  Default shunt action:                  Retain
 *Total enqueue requests:                34
 *Total requests that have waited:       8
 *Total requests failed busy:            6
 *Total requests failed locked:          2
 *Total requests timed out:              1
 *Total enqueues that were retained:     1

*NOTE: These values were reset at 15.44.39 (the last statistics interval collection)

                                              OWNER  /  WAITER
                                       NQEA   Tran Tran                    Lifetime Hash
         Enqueue Name          Len Sta Address   Id   Num     Local Uowid   Uow Tsk  Indx
------------------------------ --- --- -------- ---- ----- ---------------- --- ---  ----
Q007TOQ                          9 Act 052C4580 TDWR 00356 A8EBC70A53A4BC82   1   0   13
Q002FROMQ                        9 Act 053D0880 TDRD 00435 A8EBD91A57D9B7D2   2   0   24
                         Waiter  :    0540BBC0 TDRD 00467 A8EBDAC692BB7C10   0   1   24
                         Waiter  :    0537CE70 TDDL 00512 A8EBDAE6FF0B56F2   1   0   24
Q007FROMQ                        9 Act 0540CC80 ENQY 00217 A8EBB7FE23067C44   0   1   51
                         Waiter  :    0538F320 ENQY 00265 A8EBBF0846C00FC0   0   1   51
                         Waiter  :    0518C5C0 ENQY 00322 A8EBC393B90C66D8   0   1   51
Q002TOQ                          9 Ret 0520B260 ---- ----- A8EBD82AFDA4CD82   1   0   53
Q009FROMQ                        9 Act 0540A140 TDRD 00366 A8EBC84D3FF80250   1   0   62
```

Figure 23. Example system dump, showing summary information for the TDNQ enqueue pool

In the table at the bottom of Figure 23 on page 133, each enqueue in the pool appears on a new line. If the enqueue has waiters, they are displayed in order on subsequent lines. Waiters are identified by the string `Waiter`. The meanings of the table headings are:

Enqueue Name
The string that has been enqueued upon. Normally, up to 30 characters of the name are displayed; however, the summary reports for file control and address enqueue pools format the enqueue name differently:

- File control uses six enqueue pools for its various types of lock. Each enqueue contains the address of a control block (for example, DSNB, FCTE) in its first four bytes. If the enqueue is a record lock, this is followed by the record identifier.

  Depending upon the type of the data set or file, the remainder of the enqueue name could, for example, be an RRN in an RRDS, or a record key in a KSDS data set. In the summary, the remainder of the enqueue name is displayed in both hex and character formats. This takes up two summary lines instead of one.
- The summary reports for the EXECADDR and KCADDR enqueue pools display the enqueue name in hexadecimal format. This is because the enqueue request was made on an address.
- IPIC communication can use more than one socket for an IPCONN when communicating with another system. For each of these sockets, there is an IPIC session set (ISSS) control block and an associated enqueue name.

  The enqueue name is a character string that is composed of the first four characters of the IPCONN name followed by 'ISSS*nnnx*' where *nnn* is an index for a particular IPIC session set (ISSS) and *x* is the character S or C.

Len
The length of the enqueue name.

Sta
The state that the enqueue is held in. This field contains either:

Act
The enqueue is held in active state—that is, other transactions are allowed to wait on the enqueue.

Ret
The enqueue is held in retained state—that is, other transactions are not allowed to wait on the enqueue. Typically, this is because the enqueue is owned by a shunted unit of work.

NQEA Address
The address of the NQEA corresponding to the enqueue owner or waiter. The NQEA contains the full enqueue name if it was too large to display fully.

TranId
The transaction identifier of the enqueue owner or waiter. If the enqueue is owned by a shunted UOW, this field contains '----'.

TranNum
The task number of the enqueue owner or waiter. If the enqueue is owned by a shunted UOW, this field contains '-----'.

Local Uowid
The local UOW identifier of the enqueue owner or waiter.

Uow Lifetime
For an enqueue owner, the number of times the enqueue is owned with UOW lifetime. For an enqueue waiter, whether the waiter has requested the enqueue for the lifetime of the UOW.

Tsk Lifetime
For an enqueue owner, the number of times the enqueue is owned with task lifetime. For an enqueue waiter, whether the waiter has requested the enqueue for the lifetime of the task.

Hash Indx
An index into the pool's internal hash table.

EXEC CICS ENQ waits

These waits are a particular type of enqueue wait. They occur when an application issues an EXEC CICS ENQ command to acquire an enqueue on a resource, and another task already holds an enqueue on it.

- A resource name of EXECADDR indicates that the LENGTH option of the EXEC CICS ENQ command was omitted - that is, the RESOURCE option supplied the address of the resource to be enqueued upon.
- A resource name of EXECSTRN indicates that the LENGTH option of the EXEC CICS region ENQ command was specified - that is, the RESOURCE option supplied the name of the resource to be enqueued upon.
- A resource name of EXECPLEX indicates that the LENGTH option of the EXEC CICS sysplex ENQ command was specified - that is, the RESOURCE option supplied the name of the resource to be enqueued upon.

You can use the **CEMT INQUIRE UOWENQ** command to discover the owner of the enqueue that the suspended task is waiting on providing the owner is on the same region. This cannot detect owners on other regions. For EXECADDR type waits, to display the address of the resource specified on the EXEC CICS ENQ command you need to use the hexadecimal display option of CEMT.

For detailed information about the **EXEC CICS ENQ** command, see ENQ.

Investigating interval control waits

If you have a task that is not running and interval control seems to be involved, you can use this information to understand the possible causes.

About this task

The following is a list of possible causes, and suggestions to consider before you carry out a detailed investigation. If these do not give you enough information in order to solve the problem, go to "Finding the reason for a DELAY request not completing" on page 135 for further guidance. If, in the course of your preliminary investigations, you find that the task is waiting because the terminal where it is due to start is unavailable, turn to "Investigating terminal waits" on page 118.

Procedure

- A terminal task that should have been initiated with an **EXEC CICS START** command did not start when you expected it to. **CEMT INQ TASK** does not recognize the task, because it has not yet been attached.

  One approach is to identify the terminal where the subject task should have started, and see if that terminal is unavailable. You can use **CEMT INQ TERMINAL** to find the status of the terminal.
- You have found that 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.

  a) Find the target TERMID for other tasks issuing **EXEC CICS START** commands to supply more data.

  The resource name gives you the name of the terminal running the task in the ICGTWAIT and therefore the target TERMID.

  b) If there are no tasks in the system that would issue START commands for this TERMID, you need to determine whether this is reasonable.

  c) 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.

  d) Look at the deadlock timeout 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, 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.

  a) 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, that suggests a possible CICS error.

  b) Consider 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:

  a) Check to make sure that the interval or time specified on the POST was what you intended.

  b) If it is, check to see whether the ECB being waited on was posted. If it has been posted, that indicates a possible CICS error.

Results

If none of the simple checks outlined here help you to solve the problem, read the following information.

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.

Before you begin

About this task

Procedure

1. Take a system dump, and format it using the keywords CSA, ICP, and AP.

   These get you 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.
2. Locate field CSATODTU in the CSA and make a note of its value.

   It is the current CICS time of day in internal 'timer units'.
3. 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 its value.

Results

If the value of ICEXTOD is greater than CSATODTU, the ICE has not yet reached the expiry time. The possible explanations are:

- 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 the value of 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 the value of 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

Follow this procedure to use trace to find out why tasks are waiting on interval control.

Before you begin

Before using trace to find out why your task is waiting on interval control, select an appropriate trace destination and set up the right tracing options.

About this task

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. The sort of trace entries that you can expect in normal operation are shown in the examples below. 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.

Procedure

1. Use the CETR transaction to set up the following tracing options:
   a) Specify special tracing for the level 2 trace points for components DS (dispatcher domain), TI (timer domain), and IC (interval control program).
   b) 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.
   c) Set the main system trace flag off, to turn off all standard tracing.

      This helps minimize the number of trace entries not connected with the problem.
   d) Make sure that auxiliary tracing is active
2. Set the transaction running. When the problem appears, format the auxiliary trace data set and either print it or view it online.
3. Analyze the trace, using the examples as a guide.
   a) Figure 24 on page 137 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 24. Trace entries following **EXEC CICS DELAY INTERVAL**(000003) invocation*

      i) 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.

      ii) 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.

   b) Look next for an entry with point ID DS 0004 showing your task being suspended, as in Figure 25 on page 138.

      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 25. Trace entries showing interval calculation and task suspension

i) 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 in z/Architecture Principles of Operation.

   If you do the calculation, you 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.

ii) 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.

iii) 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 INQ TASK for tasks that are waiting on interval control.

c) Get some trace entries recording 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.

d) 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 26 on page 139.

```
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 26. Trace entries showing your task being resumed

i) Trace point **AP F322** is used to report that system task APTIX has been resumed. APTIX has the job of â€œwaking upâ€ 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.

ii) 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.

iii) 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 runâ€”it has only been made â€œdispatchableâ€. For example, it still needs to wait for a TCB to become available.

e) 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 27 on page 139.

```
aAP 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 27. Trace entries showing satisfactory conclusion of the DELAY request

i) 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.

ii) Trace point **AP 00E1** is on EXIT from the EXEC interface program. This is shown by bits 0â€“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.

Results

When you look at your own trace table, be concerned principally with 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 “Dealing with storage violations” on page 231. If you think the error is in CICS code, contact the IBM Support Center.

Investigating file control waits

Most file control waits are associated with resource types starting with the characters FC. Some are associated with resource type ENQUEUE, but ENQUEUE is not used exclusively for file control waits.

About this task

Table 13 on page 140 lists the identifiable resource types associated with file control waits, with all the possible reasons for waits, and whether they occur for files accessed in RLS mode, non-RLS mode, or both.

Table 13. Resource types for file control waits

Resource	Description	RLS or non-RLS access mode
CFDTWAIT	The task is waiting for a request to the CFDT server to complete.	N/A. The wait is caused by access to a coupling facility data table.
CFDTPOOL	The task is waiting for a CFDT "maximum requests" slot to become available.	N/A. The wait is caused by access to a coupling facility data table.
CFDTPOOL	The task is waiting for a CFDT "locking request" slot to become available.	N/A. The wait is caused by access to a coupling facility data table.
ENQUEUE	The task is waiting for a lock on a file or data table. See “Resource type ENQUEUE - waits for locks on files or data tables” on page 150.	Non-RLS
FCACWAIT	CICS is waiting for the last RLS file to close after an SMSVSAM failure.	RLS
FCBFSUSP	The task is waiting for a VSAM buffer.	Non-RLS
FCCAWAIT	CICS is waiting on a VSAM control ACB request.	RLS
FCCFQR	CICS is waiting for the SMSVSAM server to notify CICS of a new quiesce request	RLS
FCCFQS	CICS is waiting for a user task to issue a new quiesce request.	RLS
FCCRSUSP	CICS is waiting for last RLS control ACB request to complete during clean up after SMSVSAM failure.	RLS
FCDWSUSP	The task is waiting for VSAM to complete update processing.	Non-RLS
FCFRWAIT	The task is waiting for a file to finish closing.	Both
FCFSWAIT	The task is waiting to change the state of a file.	Both

Table 13. Resource types for file control waits (continued)		
Resource	**Description**	**RLS or non-RLS access mode**
FCIOWAIT	The task is waiting for I/O on a disk volume.	Non-RLS
FCIRWAIT	The task is waiting for the recoverable file control environment to be rebuilt.	Both
FCPSSUSP	The task is waiting for a private string.	Both
FCQUIES	The task is waiting for a quiesce request to complete.	RLS
FCRAWAIT	The task is waiting for file control to process non-recoverable requests at CICS restart.	Both
FCRBWAIT	The task is waiting for file control to process recoverable requests at CICS restart.	Both
FCRDWAIT	The task is waiting for a drain of the RLS control ACB to complete.	RLS
FCRPWAIT	The task is waiting for file control initialization to complete.	RLS
FCRRWAIT	The task is waiting for a dynamic RLS restart to complete.	RLS
FCRVWAIT	The task is waiting for I/O on a disk volume.	RLS
FCSHUTIM	An immediate shutdown has been invoked. Any task attempting to invoke VSAM will be put into a permanent wait.	Both
FCSRSUSP	The task is waiting for a shared resource string.	Non-RLS
FCTISUSP	The task is waiting for a VSAM transaction ID.	Non-RLS
FCXCSUSP or FCXDSUSP	The task is waiting for exclusive control of a VSAM control interval.	Non-RLS

The implications of waits on any of these file control resource types are dealt with in the sections that follow.

Resource type CFDTWAIT - wait for CFDT request to complete

If you have a task waiting on resource type CFDTWAIT, the task is waiting on the coupling facility data table server for a file control request to complete.

Requests to the CFDT server are normally processed synchronously. Therefore, this wait could indicate that:

- There is a high level of activity to the CFDT server
- The server is processing a request for a record that is longer than 4K bytes
- The task has issued a request for a record that is currently locked by another task within the sysplex.

Waiting on this resource can occur only for a file defined to access a coupling facility data table.

Resource type CFDTPOOL - wait for CFDT a request slot

If you have a task waiting on resource type CFDTPOOL, the task is waiting for a free slot within the maximum requests limit to become available.

CICS places a limit on the number of requests that a region can have running simultaneously in a coupling facility data tables server. This limit is known as the "maxreqs" limit, and it avoids overloading the coupling facility. If the number of requests currently running in the server for a CICS region has reached this limit, a request waits until one of the other requests completes.

Waiting on this resource can occur only for a file defined to access a coupling facility data table.

Resource type CFDTLRSW - wait for CFDT locking request slot

If you have a task waiting on resource type CFDTLRSW, the task is waiting for a free locking request slot to become available.

CICS places a limit on the number of locking requests (that is, requests that might acquire record locks) that a region can have simultaneously running in a coupling facility data table server. This limit is known as the locking request slot (LRS) limit, and it avoids tasks that hold locks from preventing other coupling facility data table accesses. If the number of locking requests currently running in the server for a CICS region has reached the LRS limit, this request waits for one of the locking requests to complete.

Waiting on this resource can occur only for files defined to access a coupling facility data table, and only for record access requests that could potentially require a lock.

Resource type FCACWAIT & FCCRSUSP - wait for SMSVSAM clean up

The only task that can wait on resource types FCACWAIT and FCCRSUSP is CSFR, the task that performs clean up after failure of the SMSVSAM server.

This is the server that CICS file control uses for any VSAM request it issues in RLS mode. clean up after SMSVSAM failure is in two stages.

1. Wait for VSAM to reject any file requests that were in-flight at the time of the server failure. When all these active file requests have been rejected, CSFR cleans up CICS state by issuing a CLOSE request against every file open in RLS mode. When the last CLOSE request has completed, the first stage of clean up is complete.

   If CSFR is waiting for this first stage of clean up to complete, it is waiting on resource type FCACWAIT.
2. Wait for VSAM to reject any system requests issued against the SMSVSAM control ACB, and then unregister the control ACB.

   If CSFR is waiting for this second stage of clean up to complete, it is waiting on resource type FCCRSUSP.

FCACWAIT and FCCRSUSP are RLS-related waits only.

Resource type FCBFSUSP - waits for VSAM buffers

If your task is waiting on resource type FCBFSUSP, 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 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 LSRPOOL for both base and paths.

Waiting on this resource can occur only for files accessed in non-RLS mode.

Resource type FCCAWAIT - waits on the SMSVSAM control ACB

If your task is waiting for resource type FCCAWAIT, it means that the task is waiting within VSAM for a request issued against the RLS control ACB to complete. The control ACB is used for requests that are not issued against any specific file.

For example, requests to:

- Release locks
- Convert locks to retained status
- Quiesce data sets.

If the request is to quiesce a data set, CICS is waiting for other CICS regions that access the data set to respond to the quiesce request. In other cases, the request should complete quickly. Failure to complete quickly could indicate a delay within the VSAM lock manager.

Waits on this type of resource can occur only for files accessed in RLS mode.

Resource type FCCFQR - wait for SMSVSAM server notification

The system task CFQR can wait on resource type FCCFQR. This is the normal state for this task and means that the task is waiting on an ECB for more work.

New work is created for the task when CICS receives a quiesce request from its SMSVSAM server through the CICS RLS quiesce exit program, DFHFCQX. SMSVSAM drives the CICS RLS quiesce exit, which creates a control block for the request and posts the CFQR task to notify it of the request''s arrival.

Resource type FCCFQS - wait for user task to issue quiesce

The system task CFQS can wait on resource type FCCFQS. This is the normal state for this task and means that the task is waiting on an ECB for more work.

New work is created for this system task when a user task issues a quiesce request (for example, issues an EXEC CICS SET DSNAME(...) QUIESCED WAIT command). The user request is processed by CICS module DFHFCQI, which creates a control block for the request and posts the CFQS task to notify it of the request's arrival.

Resource type FCDWSUSP - wait for VSAM to complete update processing

If your task is waiting on resource type FCDWSUSP, it has received a VSAM response which might indicate that your task is trying to read a record while this record is being updated.

Depending on the VSAM response code:

- The read is using a VSAM path while this record is being updated by another request. This other request is updating the record either using the base or 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.
- A concurrent write to the end of the data set is incomplete.

This is a transient condition. CICS waits for all current update operations for this VSAM data set to complete and retries the request twice. 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') or (X'089C') in bytes 2 and 3 of EIBRCODE. These identify the cause of the ILLOGIC response.

Waiting on this resource can occur only for files accessed in non-RLS mode.

Resource type FCFRWAIT - wait for file state changes

If your task is waiting on resource type FCFRWAIT, it means that an implicit open is being performed on a file and that file is found to be closing.

This behavior can happen, for example, when an implicit open is being performed on a file and that file is found to be closing. In that situation the task performing the implicit open waits in an FCFRWAIT for the file to finish closing at which time it attempts to open the file again.

Only one task at a time waits on FCFRWAIT. If any other tasks attempt to change the state of the same file, they are suspended on resource type ENQUEUE. See "Task control waits" on page 188.

Waiting on this resource can occur for files accessed in both RLS and non-RLS mode.

Resource type FCFSWAIT - wait 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, but another task is still using the file.

This can happen, for example, if a long-running transaction, possibly conversational, is using a recoverable file. The file cannot be closed until the updates made by the transaction have been committed; that is, the transaction has issued a syncpoint. In such a case, consider changing the programming logic so that intermediate syncpoints are issued.

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 ENQUEUE. See "Task control waits" on page 188.

Waiting on this resource can occur for files accessed in both RLS and non-RLS mode.

Resource type FCIOWAIT - wait for VSAM I/O (non-RLS)

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.

For example, VSAM uses MVS RESERVE volume locking, and it is likely that another job has at present got the lock on the volume. See if there are any messages on the MVS console to explain the error.

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 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 types FCXCSUSP, FCXDSUSP, FCXCPROT, and FCXDPROT - VSAM exclusive control waits" on page 147.

Waiting on this resource can occur only for files accessed in non-RLS mode.

Resource type FCIRWAIT - wait for FC environment to be rebuilt

During CICS initialization, on a warm or emergency restart, file control must wait for the recoverable file control environment to be rebuilt before performing any restart actions for recoverable files.

DFHFCIR is the module that rebuilds the recoverable file control environment, and the file control initialization task waits on resource type FCIRWAIT.

Because this wait occurs during CICS initialization, you should not be able to see a task waiting on this resource.

Resource types FCPSSUSP and FCSRSUSP - waits for VSAM strings

If your task is waiting on either of resource types FCPSSUSP or FCSRSUSP, it means that it cannot get a VSAM string. FCPSSUSP 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.

For non-RLS mode, the number of strings defined for a VSAM data set (STRINGS parameter in the FILE resource definition) determines how many tasks can use the data set concurrently. STRINGS can have a value in the range 1–255. For RLS mode, strings are automatically allocated as needed up to a maximum of 1024. When all the strings are in use, any other task wanting to access the data set must wait until a string has been released.

The CICS monitoring facility provides performance data for the VSAM string wait time for each user task. The performance data field 427, FCVSWTT, in the DFHFILE group, shows the elapsed time in which the task waited for a VSAM string. 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 STARTBR or READNEXT and then enters a wait for terminal input without issuing an 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 READ UPDATE or WRITE MASSINSERT is outstanding over a wait for terminal input.

Waiting on this resource can occur for files accessed in both RLS and non-RLS mode.

Resource type FCQUIES - wait for a quiesce request to complete

If your task is waiting on resource type FCQUIES, the task is waiting for the completion of a quiesce command it has issued for a data set.

For example, an EXEC CICS SET DSNAME(...) QUIESCED WAIT command. The command generates an FCQSE containing the request and passes this into the CFQS task. The CFQS task posts the user task when the request is completed. The resource name gives the hexadecimal address of the FCQSE control block.

Resource type FCRAWAIT - file control to process non-recoverable requests

A non-recoverable file control request waits on resource type FCRAWAIT if file control has not completed the actions required to allow the processing of non-recoverable work. These actions include the building of FCT entries.

You do not see a task waiting on this resource type, because this wait occurs during CICS initialization.

Waits on this resource type can occur for files accessed in both RLS and non-RLS mode.

Resource type FCRBWAIT - file control to process recoverable requests

A recoverable file control request waits on resource type FCRBWAIT if file control has not completed the actions required to allow the processing of recoverable work. These actions include the rebuilding of non-RLS enqueues and restarting RLS access.

You do not see a task waiting on this resource type, because this wait occurs during CICS initialization.

Waiting on this resource can occur for files accessed in both RLS and non-RLS mode.

Resource type FCRDWAIT - wait for a drain of the RLS control ACB

If a task is waiting on resource type FCRDWAIT, it is waiting for completion of the drain of the RLS control ACB following an SMSVSAM server failure.

DFHFCRD is the module that performs the drain. When the SMSVSAM server fails, CICS must drain all RLS processing, which involves:

- Disabling further RLS access
- Preventing existing tasks from issuing further RLS requests after the server becomes available again
- Closing all ACBs that are open in RLS mode.

If a system task is waiting on this resource, it means that it is waiting to perform a dynamic RLS restart to reestablish access to a restarted (new) SMSVSAM server. CICS access to the failed server must be drained before CICS can register with the new server.

If a user task is waiting on this resource, it means that it is waiting in backout processing for a drain to complete before checking whether the file being backed out is open, because drain affects the open state of the file.

The drain is carried out by the system task CSFR. This should normally complete without problems, although it may take some time if there is a large number of files to be closed. If a task is waiting on FCRDWAIT for a considerable length of time, you should check whether the CSFR task is itself in a wait and therefore failing to complete.

Resource type FCRPWAIT - wait for file control initialization to complete

If a task is waiting on resource type FCRPWAIT, dynamic RLS restart is waiting for file control initialization to complete.

DFHFCRP is the module that performs most of file control initialization processing. A dynamic RLS restart occurs when a restarted SMSVSAM server becomes available following a failure of the previous server. If this occurs during CICS initialization, dynamic RLS restart must wait for file control initialization to complete.

Because this wait occurs during CICS initialization, you should not be able to see a task waiting on this resource.

Resource Type FCRRWAIT - wait for dynamic RLS restart to complete

If a task is waiting on resource type FCRRWAIT it means that a dynamic RLS restart is waiting for an earlier dynamic RLS restart to complete.

DFHFCRR is the module that performs dynamic RLS restart processing.

A dynamic RLS restart occurs when a restarted SMSVSAM server becomes available following a failure of the previous server. If a restarted SMSVSAM server, which has caused one dynamic RLS restart, fails and becomes available again, it causes CICS to perform another dynamic RLS restart. If the first dynamic RLS restart has not finished, the second dynamic RLS restart must wait for the first to complete.

If a task is waiting in FCRRWAIT for a considerable length of time, you should check whether there is any other task performing dynamic RLS restart, which is itself in a wait and therefore failing to complete.

Resource type FCRVWAIT - wait for VSAM I/O (RLS)

If you have a task waiting on resource type FCRVWAIT, it means that the task is waiting within VSAM for I/O to take place, or is waiting for a record lock.

A wait on resource type FCRVWAIT occurs when conflicts over shared or exclusive locks are deferred internally by VSAM and not returned as an error condition to CICS. Conflicts that can cause an FCRVWAIT wait are:

- A task issues a file control READ UPDATE request for a record, for which:

  - Another task already holds an exclusive lock
  - One or more tasks hold a shared lock.
- A task issues a file control READ request with CONSISTENT or REPEATABLE integrity for a record, for which:
  - Another task already holds an exclusive lock.
  - Another task is waiting for an exclusive lock because one or more tasks may already have a shared lock, or another task has an exclusive lock.

Waiting on this resource can occur only for files accessed in RLS mode.

A task could be in an FCRVWAIT state because of a deadlock. If VSAM detects an RLS deadlock condition, it returns a deadlock exception condition to CICS, causing CICS file control to abend the transaction with an AFCW abend code. CICS also writes messages and trace entries that identify the members of the deadlock chain.

VSAM cannot detect a cross-resource deadlock (for example, a deadlock arising from use of RLS and DB2 resources) where another resource manager is involved. A cross-resource deadlock is resolved by VSAM when the timeout period expires, as defined by either the DTIMOUT or FTIMEOUT parameters, and the waiting request is timed out. In this situation, VSAM cannot determine whether the timeout is caused by a cross-resource deadlock, or a timeout caused by another transaction acquiring an RLS lock and not releasing it. In the event of a timeout, CICS writes trace entries and messages to identify the holder of the lock for which a timed-out transaction is waiting. Similarly, a task could be made to wait on another task that has an exclusive or shared lock on a record. If this second task was, itself, waiting for an exclusive lock on a resource for which the first task already has a lock, both tasks would be deadlocked.

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 5 for advice if you suspect this.
- There could be a performance problem. Guidance about dealing with performance problems is given in “Dealing with performance problems” on page 204.
- 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.

Waiting on this resource can occur only for files accessed in non-RLS mode.

Resource types FCXCSUSP, FCXDSUSP, FCXCPROT, and FCXDPROT - VSAM exclusive control waits

If your task is waiting on resource type FCXCSUSP, FCXDSUSP, FCXCPROT, or FCXDPROT, 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. Waits on these resources can occur only for files accessed in non-RLS mode.

An exclusive control wait on these resource types occurs in CICS, unlike the similar wait on FCIOWAIT, which occurs in VSAM. See “Resource type FCIOWAIT - wait for VSAM I/O (non-RLS)” on page 144.

FCXCSUSP
FCXCSUSP represents a suspend of a task that has received an exclusive control conflict response from VSAM. The task is suspended until the control interval is no longer held by another task.

A suspended task waiting on a resource type of FCXCSUSP is subject to its deadlock timeout value (DTIMOUT) and is purged upon timeout.

FCXDSUSP
FCXDSUSP represents a suspend of a task that has received an exclusive control conflict response from VSAM. The task is suspended when another task is being resumed in order to be removed from the exclusive control conflict tree.

A suspended task waiting on a resource type of FCXDSUSP is subject to its deadlock timeout value (DTIMOUT) and is purged upon timeout.

FCXCPROT and FCXDPROT
FCXCPROT and FCXDPROT are used where tasks are suspended for the same reasons above, but as a result of an exclusive control conflict where VSAM has also reported to CICS that general sphere inconsistencies exist. This means that VSAM has detected an error in the sphere's base cluster or index, or an error in the upgrade set. In such circumstances, CICS takes caution not to abend such a task but instead let another task complete and ideally resolve VSAM's concerns. CICS does not purge the task if it has been suspended for the amount of time longer than its DTIMOUT value.

In these cases, it is not advisable to purge the requests because the data set can be lift in an inconsistent state. Purge other tasks involved in the wait to allow CICS to retry the VSAM requests for those tasks with FCXCPROT and FCXDPROT waits.

The CICS monitoring facility provides performance data for the exclusive control wait time for each user task. The performance data field 426, FCXCWTT, in the DFHFILE group, shows the elapsed time in which the task waited for exclusive control. 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 "Exclusive control deadlock" on page 148.

Exclusive control deadlock

In non-RLS mode, without some means of avoiding it, a task could wait on itself for exclusive control of a VSAM control interval. If this was allowed to happen, the task would be deadlocked, and neither able to release exclusive control or reacquire it.

Similarly, a task could be made to wait on another task that 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.

CICS however, provides a mechanism to avoid exclusive control deadlock. If a task is waiting on resource type FCXCSUSP or FCXDSUSP and causing a task to wait (either itself or another task), causing a deadlock, the task abends either with abend code AFCF or AFCG at the time that it makes the request for exclusive control.

- A task that abends 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 abends with abend code AFCG would have been waiting for exclusive control of a VSAM control interval of which it has shared control.

See Transaction abend codes for more information about these abend codes.

To resolve the problem, you must determine which program caused the potential deadlock. Find out which programs are associated with the abending 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, see "How tasks can become deadlocked waiting for exclusive control" on page 149 for guidance about finding how the error might have occurred.

For further details on the redispatch of CICS tasks that were waiting for VSAM exclusive control of a control interval to become available, see File control operations.

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 accessed in non-RLS mode.

For the deadlock to occur, a transaction must first issue a **VSAM READ SEQUENTIAL** request using **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**, using **EXEC CICS READ UPDATE** and then **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**, using **EXEC CICS WRITE**.
- **VSAM WRITE SEQUENTIAL**, using **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 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 ENQUEUE - waits for locks on files or data tables

A resource type of ENQUEUE with a resource name beginning “FC” indicates that the task is waiting for a lock on a file or data table.

Table 14 on page 150 shows the type of lock that each of the “FC” resource names represents.

Table 14. Resource/pool names and lock types

Resource or pool name	Lock type
FCDSRECD	VSAM or CICS-maintained data table record
FCFLRECD	BDAM or user-maintained data table record
FCDSRNGE	KSDS key range
FCDSLDMD	VSAM load mode
FCDSESWR	ESDS write
FCFLUMTL	User-maintained data table load

Resource name FCDSRECD

A resource name of FCDSRECD indicates a wait for a record lock in a VSAM file or CICS-maintained data table.

When a transaction updates a record in a VSAM file or CICS-maintained data table, 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, if the file or data table is recoverable, 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 backed out.

If a transaction attempts to access a record that is locked by another transaction, it is suspended on resource type ENQUEUE until the lock is released. This can be a long wait, because an 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 a 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 is incorrect if the record has to be backed out.

There is some more information about read integrity in “Dealing with incorrect output” on page 210.

Resource name FCFLRECD

A resource name of FCFLRECD indicates a wait for a record lock in a BDAM file or user-maintained data table.

Neither BDAM nor user-maintained data tables 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. If the file or data table is recoverable, the lock is released at the end of the current unit of work. If the file or data table is not recoverable, the lock is released on completion of the REWRITE or UNLOCK operation.

If a second task attempts to update the same record while the first has the lock, it is suspended on resource type ENQUEUE.

Resource name FCDSRNGE

A resource name of FCDSRNGE indicates a wait for a range lock in a recoverable KSDS data set.

When a transaction issues a mass-insert WRITE request to a recoverable KSDS data set, CICS obtains exclusive control of a range of key values. This enables CICS to perform an efficient sequential write operation, while maintaining integrity. The range extends to the next higher key in the data set.

If another transaction tries to write a record in the locked key range, or delete the record at the end of the range, it is suspended until the range lock is released. The lock is released when the transaction holding it issues a syncpoint, ends the mass-insert operation by issuing an UNLOCK, or changes to a different range.

Resource name FCDSLDMD

A resource name of FCDSLDMD indicates a wait for a lock in a VSAM data set that has been opened in load mode.

When a VSAM data set is opened in load mode, only one request can be issued at a time. If a transaction issues a WRITE request while another transaction's WRITE is in progress, it is suspended until the first WRITE completes.

Resource name FCDSESWR

A resource name of FCDSESWR indicates a wait for an ESDS write lock.

For integrity reasons, WRITE requests to recoverable ESDS data sets must be serialized. When a transaction issues such a request, it holds the ESDS write lock for the time it takes to log the request, obtain a record lock, and write the data set record. If another transaction issues a WRITE request during this period, it is suspended until the ESDS lock is released. The lock is normally released when the WRITE completes, but may be held until syncpoint if the WRITE fails.

Resource name FCFLUMTL

A resource name of FCFLUMTL indicates a wait during loading of a user-maintained data table.

When loading a user-maintained data table from its source data set, this lock is used to serialize loading with application READ requests.

Investigating 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.

About this task

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:

Procedure

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:
   a) Use the CEMT transaction, submitting one or more of the following commands: **CEMT I SYS SOSABOVEBAR**, **CEMT I SYS SOSABOVELINE** or **CEMT I SYS SOSBELOWLINE**.
   b) To see if SOS has been reached too often, examine the job log, check the run statistics, or submit **CEMT I DSAS**.

   If SOS has been reached too often, take steps to relieve the storage constraints. For guidance about this, see Reducing storage stress in IBM Knowledge Center.
2. Check for messages that might indicate that there is an I/O error on a library.

If you find a message, investigate the reason why the I/O error occurred.

3. There could be an error within MVS. Has there been any sort of message to indicate this?

   If so, it is likely that you need to refer the problem to the IBM Support Center.

Investigating lock manager waits

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.

About this task

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 information on resource locks

This section describes the data that you should find if the resource locks are being managed correctly.

About this task

Procedure

1. 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.
2. Turn to the lock manager summary information.

   Figure 28 on page 153 is an example.
3. 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.
4. 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.
5. When you have all this information ready, contact the IBM Support Center and report the problem to them.

Example

```
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 28. Lock manager summary information

The following table describes each of the fields in the lock manager summary information.

Table 15. 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.
-> 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.

Table 15. Fields in the lock manager summary information (continued)	
Field	**Description**
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.

ECB “PSTDECB” - DLI code lock, PSB load I/O, or DMB load I/O

If you find that a task is waiting on ECB PSTDECB, it indicates either an error within CICS or IMS code, or some hardware fault preventing a PSB or DMB from being loaded.

If you have no evidence of a hardware fault, contact the IBM Support Center and report the problem to them.

Investigating transaction manager waits

About this task

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 has reached the maximum number of tasks allowed, or if the task is defined in a transaction class that is at its MAXACTIVE limit.

Maximum task condition waits

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

- MXT (maximum tasks in CICS system)
- MAXACTIVE (maximum tasks in transaction class)

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 TCLASS, the resource type is TCLASS, and the resource name is the name of the TCLASS 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 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 tclass.

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 tclass 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.

Example

```
==XM: TRANSACTION SUMMARY

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

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

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

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

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

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

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

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

TA03 94001 10E34800 T    No   ACT      00000000 Terminal 10E2C3F0 00000000 00000000 00000000 10114023 00000000 00000000
```

```
           10AD3D40          DF(AKCC)                  00000000 00000000 00000000 00000000 10E31120 00000000 00000000

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

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

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

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

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

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

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

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

MXT summary

The MXT summary indicates whether CICS is currently at the maximum number of tasks, showing the current number of queued and active transactions.

To check the status of an individual transaction, consult the main transaction summary (“Transaction summary” on page 155).

```
==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 could 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 DFHCTL01, DFHCTL02, and DFHCTL03 in Figure 29 on page 157).

```
==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 29. 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 terminated. **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, DFHXM0308, DFHXM0309, DFHXM0310. Transaction manager also takes a memory dump and message DFHME0116 is produced and contains the symptom string.

Resource type TRANDEF

The suspended transaction has attempted to update the transaction definition identified by the transaction ID but found it already locked by another transaction.

Resolving deadlocks in a CICS region

You can diagnose deadlocks between tasks wanting an exclusive lock on the same resource, such as a record in a non-RLS file, a recoverable transient data queue, or any resource represented by an **EXEC CICS ENQUEUE**.

About this task

Enqueue deadlocks between tasks occur when each of two transactions (say, A and B) needs an exclusive lock on a resource that the other holds already. Transaction A waits for transaction B to release the resource. However, if transaction B cannot release the resource because it, in turn, is enqueued on a resource held by transaction A, the two transactions are deadlocked. Further transactions may then queue, enqueued on the resources held by transactions A and B.

Use the following example to help you diagnose deadlocks. The scenario is that a user of task 32 complains that a terminal is locked and is unable to enter data.

Procedure

1. Use the command **CEMT INQUIRE TASK** to display the tasks in the system.
   For example, a display similar to the following might appear:

```
INQUIRE TASK
  STATUS:  RESULTS - OVERTYPE TO MODIFY
   Tas(0000025) Tra(CEMT) Fac(T773) Run Ter Pri( 255 )
      Sta(TO) Use(CICSUSER) Uow(AA8E9505458D8C01)
   Tas(0000028) Tra(TDUP) Fac(T774) Sus Ter Pri( 001 )
      Sta(TO) Use(CICSUSER) Uow(AA8E950545CAD227) Hty(ZCIOWAIT) Hva(DFHZARQ1)
   Tas(0000032) Tra(FUPD) Fac(T775) Sus Ter Pri( 001 )
      Sta(TO) Use(CICSUSER) Uow(AA8E950545DAC004) Hty(ENQUEUE ) Hva(FCDSRECD)
   Tas(0000035) Tra(FUPD) Fac(T784) Sus Ter Pri( 001 )
      Sta(TO) Use(CICSUSER) Uow(AA8E950545DBC357) Hty(ENQUEUE ) Hva(FCDSRECD)
   Tas(0000039) Tra(FUPD) Fac(T778) Sus Ter Pri( 001 )
      Sta(TO) Use(CICSUSER) Uow(AA8E97FE9592F403) Hty(ENQUEUE ) Hva(FCDSRECD)
   Tas(0000042) Tra(FUP2) Fac(T783) Sus Ter Pri( 001 )
      Sta(TO) Use(CICSUSER) Uow(AA8E97FE95DC1B9A) Hty(ENQUEUE ) Hva(FCDSRECD)
```

   Task 32 is waiting on an enqueue `Hty(ENQUEUE)`. You can also see that the task is waiting for a lock on a data set record `Hva(FCDSRECD)`. At this stage, you cannot tell which (if any) task has control of this resource.
2. Use the command **CEMT INQUIRE UOWENQ** at the same terminal.
   This command displays information about the owners of all enqueues held. More importantly, for deadlock diagnosis purposes, it displays information about the tasks waiting for the enqueues.
   A screen similar to the following might be displayed:

```
INQUIRE UOWENQ
  STATUS:  RESULTS
   Uow(AA8E9505458D8C01) Tra(CEMT) Tas(0000025) Act Exe Own
   Uow(AA8E950545CAD227) Tra(TDUP) Tas(0000028) Act Tdq Own
   Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Own
   Uow(AA8E950545DBC357) Tra(FUPD) Tas(0000035) Act Dat Wai
   Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Wai
   Uow(AA8E9505458D8C01) Tra(TSUP) Tas(0000034) Ret Tsq Own
   Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Own
   Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Wai
   Uow(AA8E97FE95DC1B9A) Tra(FUPD) Tas(0000042) Act Dat Own
```

You can see all the enqueue owners and waiters on the same region on this display. Tasks waiting for an enqueue are displayed immediately after the task that owns the enqueue. Owners and waiters on other regions are not displayed.

3. If you system is busy, you can clarify the display by displaying only those resources that the task you are interested in owns and waits for.

   This is called filtering. You add a filter to the end of the command as follows: **CEMT INQUIRE UOWENQ TASK(32)**.

```
  INQUIRE UOWENQ TASK(32)
  STATUS:  RESULTS
   Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Own
   Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Wai
```

   You can now see that task 32 owns one enqueue but is also waiting for another. This display shows one line of information per item, listing:

   - UOW identifier
   - Transaction identifier
   - Task identifier
   - Enqueue state (active, or retained)
   - Enqueue type
   - Relation (whether owner of the enqueue or waiter).

4. To see more information, press ENTER alongside the item that interests you.

   If you press ENTER alongside the first entry of the output from **CEMT INQUIRE UOWENQ TASK(32)**, a screen similar to the following might be displayed:

```
INQUIRE UOWENQ TASK(32)
RESULT
  Uowenq
  Uow(AA8E950545DAC004)
  Transid(FUPD)
  Taskid(0000032)
  State(Active)
  Type(Dataset)
  Relation(Owner)
  Resource(ACCT.CICS710.ACCTFILE)
  Qualifier(SMITH)
  Netuowid(..GBIBMIYA.IYA2T774.n......)
  Enqfails(00000000)
```

   This shows you details of the enqueue that task 32 owns.

5. Expand the second entry to display the enqueue that task 32 is waiting for:

```
INQUIRE UOWENQ TASK(32)
RESULT
  Uowenq
  Uow(AA8E950545DAC004)
  Transid(FUPD)
  Taskid(0000032)
  State(Active)
  Type(Dataset)
  Relation(Waiter)
  Resource(INDX.CICS710.ACIXFILE)
  Qualifier(SMITH)
  Netuowid(..GBIBMIYA.IYA2T774.n......)
  Enqfails(00000000)
```

Expanding the one-line display is useful because RESOURCE and QUALIFIER fields are then revealed. These identify the physical resource that is related to the enqueue. You can see, from the first entry in this example, that task 32 owns the enqueue on record identifier “SMITH” in the ACCT.CICS710.ACCTFILE data set. You can also see, from the second expanded entry, that task 32 is waiting on an enqueue - for record identifier “SMITH” in the INDX.CICS710.ACIXFILE data set.

6. Investigate why task 32 is waiting on the enqueue detailed in the second expanded entry.

   You need to find out which task owns this enqueue and why it is holding it for such a long time. You can do this by filtering the **CEMT INQUIRE UOWENQ** command with the RESOURCE and QUALIFIER options.

   a) Enter **CEMT INQUIRE UOWENQ RESOURCE(INDX.CICS**710**.ACIXFILE) QUALIFIER(SMITH)**.

      This shows the task that owns the enqueue that is being waited on.

      ```
      INQUIRE UOWENQ RESOURCE(INDX.CICS710.ACIXFILE) QUALIFIER(SMITH)
      STATUS:  RESULTS
       Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Own
       Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Wai
      ```

      This shows you that another task, task 39, owns the enqueue that task 32 is waiting on.

   b) Find out why task 39 is holding this enqueue, using the CEMT command again as a filter for task 39. Enter **CEMT INQUIRE UOWENQ TASK(39)**.

      ```
      INQUIRE UOWENQ TASK(39)
      STATUS:  RESULTS
       Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Wai
       Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Own
      ```

      This shows you that task 39 is also waiting for an enqueue.

   c) Expand the entry that indicates the waiting state. You might see a display similar to the following:

      ```
      INQUIRE UOWENQ TASK(39)
      RESULT
        Uowenq
        Uow(AA8E97FE9592F403)
        Transid(FUP2)
        Taskid(0000039)
        State(Active)
        Type(Dataset)
        Relation(Waiter)
        Resource(ACCT.CICS710.ACCTFILE)
        Qualifier(SMITH)
        Netuowid(..GBIBMIYA.IYA2T776.p.nk4..)
        Enqfails(00000000)
      ```

      This shows you that task 39 is waiting for the enqueue on record “SMITH” in the ACCT.CICS710.ACCTFILE data set. This is the enqueue that task 32 owns.

   You can now see that the deadlock is between tasks 32 and 39.

7. To confirm that your diagnosis is correct, filter by the RESOURCE and QUALIFIER of this enqueue.

   This also shows that task 35 also waits on the enqueue owned by task 32.

```
INQUIRE UOWENQ RESOURCE(ACCT.CICS710.ACCTFILE) QUALIFIER(SMITH)
STATUS:  RESULTS
 Uow(AA8E950545DAC004) Tra(FUPD) Tas(0000032) Act Dat Own
 Uow(AA8E950545DBC357) Tra(FUPD) Tas(0000035) Act Dat Wai
 Uow(AA8E97FE9592F403) Tra(FUP2) Tas(0000039) Act Dat Wai
```

You are now in a position of knowing which transaction(s) to cancel and investigate further.

Results

You can also use the **EXEC CICS INQUIRE UOWENQ** command or the **EXEC CICS INQUIRE ENQ** command in your applications. These return all the information that is available under **CEMT INQUIRE UOWENQ**. If you want to automate deadlock detection and resolution, these commands are of great benefit.

Note that **CEMT INQUIRE UOWENQ** can be used only for files accessed in non-RLS mode, because files accessed in RLS mode have their locks managed by VSAM, not by CICS. Deadlock and timeout detection for files accessed in RLS mode is also performed by VSAM.

Resolving deadlocks in a sysplex

About this task

Since sysplex-scope ENQUEUE supports deadlock timeout there should be no possibility of an unresolved deadlock across CICS systems.

- If a CICS task fails, the NQ domain releases all MVS ENQs held on behalf of that CICS task
- If a CICS system fails, MVS releases all MVS ENQs owned by that CICS region. This applies even if the reason for the CICS system failure was an MVS failure.

When there is a rogue task with enqueues held, which hangs or loops but is not subject to runaway, the entire region can halt. CICSPlex SM tries to assist in the determination of which task to purge to free-up the system. CICSPlex SM allows you to put out an alert when a task's suspend time is too long. Once this has occurred, you need to find the task causing the problem. To do this:

Procedure

1. Display the suspended task's details and determine what the suspend reason is.

   If the suspend reason is ENQUEUE, you have to find out which enqueue is being waited upon by this task.
2. Display the enqueues held and the one this task is waiting for using the UOWENQ display (uow) Browse for this UOWid).

   From this display you can get the enqueue name that this task is waiting for.
3. Display the details of this enqueue

   You are now in a position to analyze the problem to determine the cause of the problem.

Results

Resolving indoubt and resynchronization failures

About this task

For examples of how to resolve indoubt and resynchronization failures, see Troubleshooting intersystem problems.

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 separately in the following information.

Procedure

1. If CICS stalls during initialization, read “CICS has stalled during initialization” on page 162.
2. If CICS stalls during a run, read “CICS has stalled during a run” on page 162.
3. If CICS stalls during termination, read “CICS has stalled during termination” on page 164

CICS has stalled during initialization

If CICS stalls during initialization, on an initial, cold, warm, or emergency start, the first place to look is the MVS 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 a cold start, loading the GRPLIST definitions from the CSD data set can take several minutes. For large systems, the delay could be 20 minutes or more while this takes place. You can tell if this stage of initialization has been reached because you get this console message:

```
DFHSI1511 INSTALLING GROUP LIST xxxxxxxx
```

On a warm start, there may be a considerable delay while resource definitions are being created from the global catalog.

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 Writing initialization and shutdown programs .

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 MVS.

Look first on your MVS 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 MVS 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 “Dealing with loops” on page 196 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.

The way you can find out if any of these apply to your system is dealt with in the information that follows. For some of the investigations, you will need to see a system dump of the CICS region. If you do not already have one, you can request one using the MVS 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?

The system definition parameters for your system might be causing it to stall, possibly at a critical loading. Check what is specified, paying particular attention to the following items:

- The CICS maximum tasks (MXT) and transaction class (MAXACTIVE) limits. If these parameters are too low, new tasks might fail to be attached. If you think that one of these limits might be the cause of the stall, read "Are MXT or transaction class limits causing the stall?" on page 163 for advice about further investigation.
- ICV system initialization parameter, the system region exit time. If this parameter is set too high, CICS might relinquish control to the operating system for longer than intended when it has no work to do, and might give the impression of a stall.
- ICVR system initialization parameter, the runaway task time interval. If this parameter is set too high, a runaway task might stop other tasks from running for a relatively long time. The maximum ICVR value is 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 parameter 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.

For more details about the choice of these and other system definition parameters, see Interval control value parameters and Improving the performance of a CICS system.

Is the system short on storage?

Storage manager statistics and console messages can indicate that the system is short on storage.

If storage is under stress, storage manager statistics indicate that a storage stress situation has occurred. For example, check the "Times went short on storage" and "Total time SOS" statistics.

Also, if the short-on-storage (SOS) condition is caused by a suspended GETMAIN request, or if CICS cannot alleviate the situation by releasing programs with no current user and by slowing the attachment of new tasks, the following actions occur:

- A message that states that CICS is short on storage is sent to the console:
  - DFHSM0131 for storage below 16 MB
  - DFHSM0133 for storage above 16 MB but below 2 GB
  - DFHSM0606 for storage above the bar
- The storage manager statistic "Times went short on storage" is updated.

CICS can become short on storage independently in any dynamic storage area (DSA). You might see tasks suspended on any of the following resource types: CDSA, SDSA, RDSA, UDSA, ECDSA, ESDSA, ERDSA, EUDSA, ETDSA, GCDSA, GUDSA, or GSDSA.

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 Enqueue Pool Summary in the NQ section of the dump for a summary of task enqueues and resources. This section of the dump lists all enqueues in CICS. Look for any enqueues that have many tasks in a waiting state. If there are any, look for the unit of work (UOW) for which the enqueue state is active. Look to see if this UOW is waiting on a resource.

Is there an exclusive control conflict on a volume?

Some programs use MVS RESERVE to gain exclusive control of a volume, and nothing else can have access to any data set on that volume until it is released. Watch for operations involving database access, because these could indicate an exclusive control conflict on a volume.

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 the z/OS Communications Server from the MVS console. To do this, use the command `F cicsname,CEMT INQ VTAM`. Substitute the name of the CICS job for “cicsname”. You can only use this command if the MVS console has been defined to CICS as a terminal. The status returned has a value of OPEN or CLOSED.

- If the Communications Server status is OPEN, the problem could be associated with processing done in the Communications Server 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 either CICS Communications Server exit tracing or Communications Server buffer tracing. For guidance about using these techniques, see Using CICS trace.
- If the Communications Server status is CLOSED, CICS cannot use the Communications Server to perform communication functions.

Is there an MVS system logger error?

If you suspect that there may be a problem with the MVS system logger, see “Log manager waits” on page 187.

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 MVS console or the NetView® main 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 Writing initialization and shutdown programs .

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 dispatcher functions associated with the suspension and subsequent resumption of tasks are described in detail in Dispatcher domain (DS). You can use trace to see the dispatcher functions that are requested, and the values of parameters that are supplied. See “Investigating waits using trace” on page 112 .

Some of the dispatcher 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 XPI functions (by domain).

If you want guidance about using online or offline techniques to investigate waits, see “Techniques for investigating waits” on page 111.

If you already know the identity of the resource that a task is waiting for, but are not sure what functional area of CICS is involved, see Table 16 on page 166. It shows you where to look for further guidance.

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.

This section covers information on the following waits:

- “Interregion and intersystem communication waits” on page 191
- “Transient data waits” on page 192
- “CICS system task waits” on page 195
- “FEPI waits” on page 195
- “Recovery manager waits” on page 196

The resources that CICS tasks can wait for

Tasks in a CICS system can wait for various resources.

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 that are 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 for some of the resources. For each such resource, a topic reference shows you where to look for guidance about dealing with the wait. The two values in the column **Purge status** indicate whether the suspending module permits normal task purging (such as that caused by the API and CEMT purge commands) and purging caused by a deadlock timeout limit being reached. The first value indicates whether normal task purging is permitted; the second indicates whether deadlock timeout is permitted.

The remaining resources are used only by CICS system tasks. If you have evidence that a system task is waiting for such a resource, and it is adversely affecting the operation of your system, you probably need to contact your IBM Support Center. Before doing so, however, read “CICS system task waits” on page 195.

Table 16. Resources that a suspended task might wait for

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
(none)		(none)	DFHDUIO	WAIT_MVS IO	System only	“CICS system task waits” on page 195
(none)		(none)	DFHRMSL7	WAIT_MVS TIMER	System only	“CICS system task waits” on page 195
(none)		(none)	DFHZNAC	SUSPEND See note “1” on page 181	System only	“CICS system task waits” on page 195
(none)		DLCNTRL	DFHDBCT	WAIT_MVS See note “1” on page 181	System only	“CICS system task waits” on page 195
(none)		DLCONECT	DFHDBCON	WAIT_MVS OTHER_ PRODUCT	System only	“CICS system task waits” on page 195
(none)		DMWTQUEU	DFHDMWQ	SUSPEND MISC	System only	“CICS system task waits” on page 195
(none)	No, No	LMQUEUE	DFHLMLM	SUSPEND LOCK	User	“Investigating lock manager waits” on page 152
ADAPTER	No, No	FEPI_RQE	DFHSZATR	WAIT_MVS MISC	User	See note “2” on page 181

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
ALLOCATE	Yes, Yes	TCTTETI value	DFHALP	SUSPEND See note "3" on page 181	User	"Interregion and intersystem communication waits" on page 191
ALP_TERM		(none)	DFHALRC	WAIT_OLDC MISC	System only	"Recovery manager waits" on page 196
Any_MBCB	No, No	Transient data queue name	DFHTDB DFHTDRM	SUSPEND IO	User	"Transient data waits" on page 192
Any_MRCB	No, No	Transient data queue name	DFHTDB DFHTDRM	SUSPEND IO	User	"Transient data waits" on page 192
AP_INIT		ECBTCP	DFHAPSIP	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
AP_INIT		SIPDMTEC	DFHAPSIP	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
AP_INIT		TCTVCECB	DFHSII1	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
AP_INIT		ZGRPECB	DFHSII1	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
AP_QUIES		CSASSI2	DFHSTP	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
AP_QUIES		SHUTECB	DFHSTP	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
APRDR		INITIAL	DFHAPRDR	SUSPEND MISC	System only	"Recovery manager waits" on page 196
APRDR		RECOVER	DFHAPRC	SUSPEND MISC	System only	"Recovery manager waits" on page 196
AP_TERM		STP_DONE	DFHAPDM	WAIT_MVS LOCK	System only	"CICS system task waits" on page 195
AS_CHILD	Yes	(none)	DFHASAS	WAIT_MVS MISC	User	"Asynchronous services waits" on page 182
AS_ANY	Yes	(none)	DFHASAS	WAIT_MVS MISC	User	"Asynchronous services waits" on page 182
ASPARENT	Yes	(none)	DFHASAS	SUSPEND MISC	User	"Asynchronous services waits" on page 182
CCSTWAIT		VSMSTRNG	DFHCCCC	WAIT_OLDC IO	System only	"CICS system task waits" on page 195
CCVSAMWT		ASYNRESP	DFHCCCC	WAIT_MVS IO	System only	"CICS system task waits" on page 195

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
CCVSAMWT		EXCLOGER	DFHCCCC	WAIT_MVS IO	System only	"CICS system task waits" on page 195
CDB2CONN	No, No	(none)	DFHD2EX1	WAIT_MVS OTHER_ PRODUCT	User	"CICS DB2 waits" on page 185
CDB2RDYQ	No, No	Name of DB2ENTRY or pool	DFHD2EX1	WAIT_MVS OTHER_ PRODUCT	User	"CICS DB2 waits" on page 185
CDB2TCB	No, No	(none)	DFHD2EX1	WAIT_MVS OTHER_ PRODUCT	User	"CICS DB2 waits" on page 185
CDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
CFDTWAIT		File name	DFHFCDO DFHFCDR DFHFCDU	WAIT_MVS MISC WAIT_MVS MISC WAIT_MVS MISC	User	"Investigating file control waits" on page 140
CFDTPOOL		CFDT pool name	DFHFCDO DFHFCDR DFHFCDU	SUSPEND LOCK SUSPEND LOCK SUSPEND LOCK	User	"Investigating file control waits" on page 140
CFDTLRSW		CFDT pool name	DFHFCDR	SUSPEND LOCK	User	"Investigating file control waits" on page 140
CSNC		MROQUEUE	DFHCRNP	WAIT_MVS See note "1" on page 181	System only	"CICS system task waits" on page 195
DB2	No, No	LOT_ECB	DFHD2EX1	WAIT_MVS OTHER_ PRODUCT	User	"CICS DB2 waits" on page 185
DB2_INIT	Yes, Yes	(none)	DFHD2IN1	WAIT_OLDC MISC	User	"CICS DB2 waits" on page 185
DB2CDISC	Yes, Yes	Name of DB2CONN	DFHD2TM	WAIT_OLDC MISC	User	"CICS DB2 waits" on page 185
DB2EDISA	Yes, Yes	Name of DB2ENTRY	DFHD2TM	WAIT_OLDC MISC	User	"CICS DB2 waits" on page 185
DBDXEOT		(none)	DFHDXSTM	WAIT_MVS MISC	System only	"CICS system task waits" on page 195

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
DBDXINT		(none)	DFHXSTM	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
DBCTL	No, No	DLSUSPND	DFHDBSPX	WAIT_MVS OTHER_ PRODUCT	User	"DBCTL waits" on page 186
DFHAIIN		AITM	DFHAIIN1	SUSPEND MISC	System only	"CICS system task waits" on page 195
DFHCPIN		CPI	DFHCPIN1	SUSPEND MISC	System only	"CICS system task waits" on page 195
DFHPRIN		PRM	DFHPRIN1	SUSPEND MISC	System only	"CICS system task waits" on page 195
DFHPTTW	Yes	DFHPTTW	DFHPTTW	SUSPEND MISC	User	
DFHSIPLT		EARLYPLT	DFHSII1	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
DFHSIPLT		LATE_PLT	DFHSIJ1	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
DISPATCH	Yes, Yes	JVM_POOL	DFHDSDS4	SUSPEND MISC	User	"Dispatcher waits" on page 183
DISPATCH	Yes, Yes	OPENPOOL	DFHDSDS4	SUSPEND MISC	User	"Dispatcher waits" on page 183
DISPATCH		OPEN_DEL	DFHDSDS4	SUSPEND MISC	User	"Dispatcher waits" on page 183
DISPATCH	No	XMCHILD	DFHXMRU	SUSPEND MISC	User	"Dispatcher waits" on page 183
DISPATCH	No	XMPARENT	DFHXMRU	SUSPEND CONV	User	"Dispatcher waits" on page 183
DISPATCH	No	XMPARENT	DFHXMRU	SUSPEND MISC	User	"Dispatcher waits" on page 183
DMATTACH		QUIESCE	DFHDMDM	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
DS_ASSOC	Yes, No	WAIT_ASSOCIATION	DFHDSAC	SUSPEND MISC	User	"Dispatcher waits" on page 183
ECDSA		(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
ECDFQEMW	No, No	ECSUSPND	DFHECSC	WAIT_MVS MISC	System	*See note* "16" on page 182
EDF	Yes, No	DBUGUSER	DFHEDFX	SUSPEND MISC	User	"EDF waits" on page 187
EKCWAIT	No, Yes	ATCHMSUB	DFHD2STR	WAIT_OLDW MISC	User	../dfhtk0z.dita

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
EKCWAIT	No, Yes	CEX2TERM	DFHD2STP	WAIT_OLDW MISC	User	../dfhtk0z.dita
EKCWAIT	No, Yes	DTCHMSUB	DFHD2STR	WAIT_OLDW MISC	User	../dfhtk0z.dita
EKCWAIT	No, Yes	MSBRETRN	DFHD2STP	WAIT_OLDW MISC	User	../dfhtk0z.dita
EKCWAIT	No, No	SINGLE	DFHEKC	WAIT_OLDW MISC	User	"Task control waits" on page 188
ENF		NOTIFY	DFHDMENF	WAIT_MVS See note "1" on page 181	System only	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	EXECADDR	DFHNQED	SUSPEND LOCK	User	"EXEC CICS ENQ waits" on page 134
ENQUEUE	Yes, Yes	EXECSTRN	DFHNQED	SUSPEND LOCK	User	"EXEC CICS ENQ waits" on page 134
ENQUEUE	Yes, Yes	FCDSESWR	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	FCDSLDMD	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	FCDSRECD	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	FCDSRNGE	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	FCFLRECD	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	FCFLUMTL	DFHNQED	SUSPEND LOCK	User	"Investigating file control waits" on page 140
ENQUEUE	Yes, Yes	ISSSENQP	DFHNQED	SUSPEND LOCK	System or user	"Investigating enqueue waits" on page 131
ENQUEUE	Yes, Yes	JOURNALS	DFHNQED	SUSPEND LOCK	User	"Investigating enqueue waits" on page 131
ENQUEUE	Yes, Yes	KCADDR	DFHNQED	SUSPEND LOCK	System or user	"Investigating enqueue waits" on page 131

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
ENQUEUE	Yes, Yes	KCSTRING	DFHNQED	SUSPEND LOCK	System or user	"Investigating enqueue waits" on page 131
ENQUEUE	Yes, Yes	LOGSTRMS	DFHNQED	SUSPEND LOCK	User	"Investigating enqueue waits" on page 131
ENQUEUE	Yes, Yes	TDNQ	DFHNQED	SUSPEND LOCK	User	"Transient data waits" on page 192
ENQUEUE	Yes, Yes	TSNQ	DFHNQED	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
EPECQEMT	No, No	EPSUSPND	DFHEPEV	WAIT_MVS MISC	System	*See note* "13" on page 182
EPEDTBMT	No, No	EPSUSPND	DFHEPEV	WAIT_MVS MISC	System	*See note* "14" on page 182
ERDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
ESDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
EUDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
FCACWAIT		*CTLACB*	DFHFCRD	WAIT_OLDC IO	System	"Investigating file control waits" on page 140
FCBFSUSP	Yes, Yes	file ID	DFHFCVR	SUSPEND IO	User	"Investigating file control waits" on page 140
FCCAWAIT	No, No	*CTLACB*	DFHFCCA	WAIT_MVS OTHER_ PRODUCT	User	"Investigating file control waits" on page 140
FCCFQR		(none)	DFHFCQR	WAIT_MVS See note "1" on page 181	System	"Investigating file control waits" on page 140
FCCFQS		(none)	DFHFCQS	WAIT_MVS MISC	System	"Investigating file control waits" on page 140
FCCRSUSP		*CTLACB*	DFHFCRD	SUSPEND IO	System	"Investigating file control waits" on page 140

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
FCDWSUSP	Yes, Yes	File ID	DFHFCVR	SUSPEND IO	User	"Investigating file control waits" on page 140
FCFRWAIT	Yes, Yes	File ID	DFHFCFR	WAIT_OLDC MISC	User	"Investigating file control waits" on page 140
FCFSWAIT	Yes, Yes	File ID	DFHFCFS	WAIT_OLDC IO	User	"Investigating file control waits" on page 140
FCINWAIT		STATIC	DFHFCIN1	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
FCIOWAIT	No, No	File ID	DFHFCBD DFHFCVR	WAIT_MVS IO WAIT_MVS IO	User	"Investigating file control waits" on page 140
FCIRWAIT		RECOV-FC	DFHFCRP DFHFCRR	WAIT_OLDC MISC WAIT_OLDC M ISC	System only	"Investigating file control waits" on page 140
FCPSSUSP	Yes, Yes	*CTLACB* file ID file ID	DFHFCCA DFHFCRS DFHFCVR	SUSPEND IOSUSPEND IOSUSPEND IO	User	"Investigating file control waits" on page 140
FCQUIES	Yes, Yes	fcqse_ptr (hexa-decimal)	DFHFCQI	SUSPEND See note "1" on page 181	User	"Investigating file control waits" on page 140
FCRAWAIT	Yes, Yes	FC_FILE	DFHEIFC	WAIT_OLDC MISC	User	"Investigating file control waits" on page 140
FCRBWAIT	Yes, Yes	File ID	DFHFCFR	WAIT_OLDC IO	User	"Investigating file control waits" on page 140
FCRDWAIT	No, No	*CTLACB*	DFHFCRC DFHFCRR	WAIT_OLDC MISC WAIT_OLDC MISC	System or user	"Investigating file control waits" on page 140
FCRPWAIT		FC-START	DFHFCRR	WAIT_OLDC MISC	System only	"Investigating file control waits" on page 140
FCRRWAIT		*DYRRE*	DFHFCRR	WAIT_OLDC MISC	System only	"Investigating file control waits" on page 140
FCRVWAIT	No, No	File ID	DFHFCRV	WAIT_MVS OTHER_ PRODUCT	User	"Investigating file control waits" on page 140

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
FCSRSUSP	Yes, Yes	File ID	DFHFCVR	SUSPEND IO	User	"Investigating file control waits" on page 140
FCTISUSP	Yes, Yes	File ID	DFHFCVR	SUSPEND IO	User	"Investigating file control waits" on page 140
FCXCSUSP and FCXDSUSP	Yes, Yes	File ID	DFHFCVS	WAIT_OLDC IO	User	"Investigating file control waits" on page 140
FCXCPROT and FCXDPROT	No, No	File ID	DFHFCVS	WAIT_OLDC IO	User	"Investigating file control waits" on page 140
FEPRM	No, No	SZRDP	DFHSZRDP	WAIT_MVS MISC	CSZI	See note "2" on page 181
FOREVER	No, No	DFHXMTA	DFHXMTA	WAIT_MVS MISC	User	"A user task is waiting on resource type FOREVER" on page 158
ICEXPIRY		DFHAPTIX	DFHAPTIX	SUSPEND TIMER	System only	"CICS system task waits" on page 195
ICGTWAIT	Yes, Yes	Terminal ID	DFHICP	SUSPEND MISC	User	"Investigating interval control waits" on page 64
ICMIDNTE		DFHAPTIM	DFHAPTIM	SUSPEND TIMER	System only	"CICS system task waits" on page 195
ICP_INIT		(none)	DFHICRC	WAIT_OLDC MISC	System only	"Investigating interval control waits" on page 64
ICP_TERM		(none)	DFHICRC	WAIT_OLDC MISC	System only	"Investigating interval control waits" on page 64
ICP_TSWT		(none)	DFHICRC	WAIT_OLDC MISC	System only	"Investigating interval control waits" on page 64
ICWAIT	Yes, No	Terminal ID (See note "4" on page 181)	DFHICP	SUSPEND MISC	User	"Investigating interval control waits" on page 64
IRLINK	Yes, No	SYSIDNT concat-enated with session name	DFHZIS2	WAIT_MVS See note "5" on page 181	User	"Investigating terminal waits" on page 118

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
IS_ALLOC	Yes	IPCONN	DFHISAL	SUSPEND	User	"Interregion and intersystem communication waits" on page 191
IS_ERROR	No	IS_ERROQ	DFHISEM	SUSPEND	System only	"Interregion and intersystem communication waits" on page 191
IS_INPUT	No	TERMID	DFHISRR	SUSPEND	User	"Interregion and intersystem communication waits" on page 191
IS_PACE	Yes	IPCONN	DFHISSR	SUSPEND	User	"Interregion and intersystem communication waits" on page 191
IS_RECV	Yes	IPCONN	DFHISSR	SUSPEND	User	"Interregion and intersystem communication waits" on page 191
IS_SESS	Yes	TERMID	DFHISIC	SUSPEND	User	"Interregion and intersystem communication waits" on page 191
JVMTHRED	Yes	JVM server name	DFHSJTH	SUSPEND MISC	System or user	See note "15" on page 182
KCCOMPAT	No, No	CICS	DFHXCPA	WAIT_OLDC LOCK	User	"Task control waits" on page 188
KCCOMPAT	No, No	LIST	DFHXCPA	WAIT_OLDW MISC	User	"Task control waits" on page 188
KCCOMPAT	No, No	SINGLE	DFHXCPA	WAIT_OLDW MISC	User	"Task control waits" on page 188
KCCOMPAT	Yes, No	SUSPEND	DFHXCPA	SUSPEND MISC	User	"Task control waits" on page 188
KCCOMPAT	Yes, No	TERMINAL	DFHXCPA	SUSPEND MISC	User	"Task control waits" on page 188 and "Investigating terminal waits" on page 118
LATE_PLT		DFHSIPLT	DFHSIPLT	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
LG_DEFER	No, No	Journal name	DFHL2SRC	SUSPEND IDLE	User	"Log manager waits" on page 187

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
LGDELALL	No, No	Journal name	DFHL2HS4	WAIT_MVS IO	User	"Log manager waits" on page 187
LGDELRAN	No, No	Journal name	DFHL2HS5	WAIT_MVS IO	User	"Log manager waits" on page 187
LGENDBLK	No, No	Journal name	DFHL2HS9	WAIT_MVS IO	User	"Log manager waits" on page 187
LGENDCRS	No, No	Journal name	DFHL2HSJ	WAIT_MVS IO	User	"Log manager waits" on page 187
LG_FORCE	Yes, No	Journal name	DFHL2SRC	SUSPEND MISC	User	"Log manager waits" on page 187
LGFREVER	No, No	DFHLOG	DFHL2SLE	SUSPEND IDLE	User	"Log manager waits" on page 187
LGHARTBT	No, No	LG_MGRST	DFHLGHB	SUSPEND TIMER	System only	"How CICS checks for the availability of the MVS logger" on page 242
LGREDBLK	No, No	Journal name	DFHL2HS8	WAIT_MVS IO	User	"Log manager waits" on page 187
LGREDCRS	No, No	Journal name	DFHL2HSG	WAIT_MVS IO	User	"Log manager waits" on page 187
LGSTRBLK	No, No	Journal name	DFHL2HS7	WAIT_MVS IO	User	"Log manager waits" on page 187
LGSTRCRS	No, No	Journal name	DFHL2HS6	WAIT_MVS IO	User	"Log manager waits" on page 187
LGWRITE	No, No	Journal name	DFHL2HSF	WAIT_MVS IO	User	"Log manager waits" on page 187
MBCB_xxx *(See note "6" on page 181)*	No, No	Transient data queue name	DFHTDB DFHTDRM	SUSPEND IO	User	"Transient data waits" on page 192
MPDFQEMW	No, No	MPSUSPND	DFHMPSC	WAIT_MVS MISC	System	*See note* "17" on page 182
MQseries	Yes, Yes	GETWAIT	DFHMQTRU	WAIT_MVS OTHER_PRODUCT	User	"IBM MQ waits" on page 186
MRCB_xxx *(See note "6" on page 181)*	No, No	Transient data queue name	DFHTDB DFHTDRM	WAIT_MVS IO	User	"Transient data waits" on page 192
MXT	No, No	XM_HELD	DFHXMAT	See note "7" on page 181	User	"Maximum task condition waits" on page 154
PIIS	Yes, Yes	RZCBNOTI	DFHPIIS	SUSPEND MISC	System	See note "8" on page 181

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
PROGRAM	Yes, Yes	program ID	DFHLDDMI	SUSPEND LOCK	User	"Investigating loader waits" on page 151
PROGRAM	Yes, Yes	program ID	DFHPGEX DFHPGIS DFHPGLD DFHPGLE DFHPGLK DFHPGLU DFHPGPG DFHPGRP DFHPGXE	SUSPEND MISC	User	
RCP_INIT		(none)	DFHAPRC	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
RDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
RMI		DFHERMRS	DFHERMRS	WAIT_MVS TIMER	System only	"Recovery manager waits" on page 196
RMCLIENT	No, No	Client name	DFHRMCIC	SUSPEND MISC	User	"Recovery manager waits" on page 196
RMUOWOBJ	No, No	LOGMOVE EXISTENC	DFHRMUO DFHRMUW DFHRMUWJ DFHRMUWS DFHRMU1U DFHRMUO DFHRMUW DFHRMUWL DFHRMUWP DFHRMUWQ DFHRMU1D DFHRMU1K	SUSPEND LOCK	User	"Recovery manager waits" on page 196
RRMSEXIT	Yes, Yes	GET_CLIENT_REQ	DFHRXUW	WAIT_MVS IDLE	User	"RRMS waits" on page 188
RRMSEXIT	No, No	NOTIFICATION	DFHRXDM	WAIT_MVS IDLE	System	"RRMS waits" on page 188
RRMSEXIT	No, No	RESYNC	DFHRXDM	WAIT_MVS IDLE	System	"RRMS waits" on page 188
RRMSEXIT	Yes, Yes	SYNCPOINT	DFHRXUW	WAIT_MVS IDLE	User	"RRMS waits" on page 188

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
RZRSTRIG	Yes, Yes	(none)	DFHRZSO DFHRZTA	SUSPEND MISC	Misc	See note "11" on page 182
RZRSTRAN	Yes, Yes	(none)	DFHRZSO DFHRZTA	SUSPEND MISC	Misc	See note "12" on page 182
SDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
SOCKET	Yes	STE	DFHSOCK	SUSPEND MISC	System only	
SOCKET	Yes	MAXSOCKETS	DFHSOEC	WAIT_MVS IDLE	System only	
SOCKET	Yes	SOCLOSE	DFHSOSO	SUSPEND MISC	System or user	
SOCKET	Yes	RECEIVE	DFHSOEC	WAIT_MVS IDLE	System or user	
SOCKET	Yes	SEND	DFHSOEC	WAIT_MVS IDLE	System or user	
SOCKET	Yes	MISCELANEOUS	DFHSOEC	WAIT_MVS IDLE	System or user	
SODOMAIN	Yes	CSOL_REG	DFHSODM	WAIT_MVS MISC	System only	
SODOMAIN	Yes	SO_LTEPTY	DFHSODM	WAIT_MVS MISC	System only	
SODOMAIN	No	SO_LISTN	DFHSOLS	WAIT_MVS MISC	System only	
SODOMAIN	No	SO_NOWORK	DFHSOLS	WAIT_MVS MISC	System only	
SODOMAIN	No	SO_LTERDC	DFHSORD	WAIT_MVS MISC	System or user	
STP_TERM		(none)	DFHAPRC	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
SMPRESOS		(none)	DFHSMSY	WAIT_MVS TIMER	System only	"CICS system task waits" on page 195
SMSYSTEM		(none)	DFHSMSY	SUSPEND TIMER	System only	"CICS system task waits" on page 195
SMSYRE		SMVA_ECB	DFHSMSY	WAIT_MVS MISC	System only	"CICS system task waits" on page 195
SUCNSOLE		WTO	DFHSUWT	WAIT_MVS MISC	System only	"CICS system task waits" on page 195

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
TCLASS	No, No	tclass name	DFHXMAT	See note "9" on page 182	User	"Checking log stream status" on page 249
TCLASS	Yes, Yes	tclass name	DFHXMCL	SUSPEND LOCK	User	
TCP_NORM		DFHZDSP	DFHZDSP	WAIT_OLDW See note "1" on page 181	System only	"CICS system task waits" on page 195
TCP_SHUT		DFHZDSP	DFHZDSP	WAIT_OLDW MISC	System only	"CICS system task waits" on page 195
TCTVCECB		ZC_ZGRP	DFHZGRP	WAIT_OLDC MISC	System only	"CICS system task waits" on page 195
TDEPLOCK	No, No	Transient data queue name	DFHTDA	SUSPEND LOCK	User	"Transient data waits" on page 192
TD_INIT	No, No	DCT	DFHTDA	SUSPEND MISC	User	"Transient data waits" on page 192
TDIPLOCK	No, No	Transient data queue name	DFHTDB	SUSPEND LOCK	User	"Transient data waits" on page 192
TD_READ	No, No	Transient data queue name	DFHTDB	SUSPEND LOCK	User	"Transient data waits" on page 192
TIEXPIRY		DS_NUDGE	DFHTISR	SUSPEND TIMER	System only	"CICS system task waits" on page 195
TRANDEF	Yes, Yes	Transaction id	DFHXMDD DFHXMQD	SUSPEND LOCK	User	"Resource type TRANDEF" on page 158
TSAUX	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSBUFFER	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSEXTEND	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSIO	No, No	(none)	DFHTSAM	WAIT_MVS IO	User	"Investigating temporary storage waits" on page 129
TSIOWAIT		DFHTEMP	DFHTSDM	WAIT_MVS IO	System only	"Investigating temporary storage waits" on page 129

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
TSMAINLM	Yes, Yes	DFHTSSQ	DFHTSSQ	SUSPEND MISC	System only	"Investigating temporary storage waits" on page 129
TSPOOL	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSQUEUE	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSSHARED	Yes, Yes	Temporary storage queue name	DFHTSSH	WAIT_MVS MISC	User	"Investigating temporary storage waits" on page 129
TSSTRING	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
TSWBUFFR	Yes, Yes	Temporary storage queue name	DFHTSWQ	SUSPEND LOCK	User	"Investigating temporary storage waits" on page 129
UDSA	Yes, Yes	(none)	DFHSMSQ	SUSPEND MISC	User	"Investigating storage waits" on page 128
USERWAIT	Yes, Yes or No, No	Supplied by application	DFHEIQSK	WAIT_MVS MISC WAIT_OLDW MISC	User	"Task control waits" on page 188
USERWAIT	Yes, Yes	CDB2TIME	DFHD2EX2	WAIT_OLDW MISC	System	../dfhtk0z.dita
USERWAIT	Yes, Yes	DB2START	DFHD2EX2	WAIT_MVS MISC	System only	../dfhtk0z.dita
WBALIAS	No, No	Target Transid	DFHWBXN	SUSPEND MISC	User	"CICS Web waits" on page 196
WEB_ECB		DFH_STATE_ TOKEN	DFHWBST	WAIT_MVS TIMER	System or User	
WMQ_INIT	Yes, Yes	(none)	DFHMQIN1	WAIT_OLDC MISC	User	"IBM MQ waits" on page 186
WMQCDISC	Yes, Yes	Name of MQCONN	DFHMQTM	WAIT_OLDC MISC	User	"IBM MQ waits" on page 186
XRGETMSG		Message queue name	DFHWMQG	WAIT_MVS See note "1" on page 181	System only	"CICS system task waits" on page 195

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
XRPUTMSG	Yes, Yes	Message queue name	DFHWMQP	WAIT_MVS MISC	User	
ZC	Yes, No	DFHZCRQ1	DFHZCRQ	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZC	Yes, No	DFHZEMW1	DFHZEMW	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZC	Yes, No	DFHZIS11	DFHZIS1	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZC	Yes, No	DFHZRAQ1	DFHZRAQ	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZC	Yes, No	DFHZRAR1	DFHZRAR	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZC_ZCGRP		ZSLSECB	DFHZCGRP	WAIT_MVS MISC	System only	"SNA LU control waits" on page 190
ZC_ZGCH	No, No	CHANGECB	DFHZGCH	WAIT_MVS MISC	User	"SNA LU control waits" on page 190
ZC_ZGIN	No, No	INQ_ECB_	DFHZGIN	WAIT_MVS MISC	User	"SNA LU control waits" on page 190
ZC_ZGRP		PSINQECB	DFHZGRP	WAIT_MVS MISC	System only	"SNA LU control waits" on page 190
ZC_ZGRP		PSOP1ECB	DFHZGRP	WAIT_MVS MISC	System only	"SNA LU control waits" on page 190
ZC_ZGRP		PSOP2ECB	DFHZGRP	WAIT_MVS MISC	System only	"SNA LU control waits" on page 190
ZC_ZGUB		PSUNBECB	DFHZGUB	WAIT_OLDC MISC	System only	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZARQ1	DFHZARQ	SUSPEND See note "10" on page 182	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZARL1	DFHZARL	SUSPEND See note "10" on page 182	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZARL4	DFHZARL	SUSPEND See note "10" on page 182	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZARR1	DFHZARR1	SUSPEND See note "10" on page 182	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZARER	DFHZARER	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZERH1	DFHZERH	SUSPEND CONV	User	"SNA LU control waits" on page 190
ZCIOWAIT	Yes, No	DFHZERH2	DFHZERH	SUSPEND CONV	User	"SNA LU control waits" on page 190

Table 16. Resources that a suspended task might wait for (continued)

Resource type	Purge status	Resource name	Suspending module	DSSR call and WLM wait type	Task	Where to look next
ZCIOWAIT	Yes, No	DFHZERH3	DFHZERH	SUSPEND CONV	User	"SNA LU control waits" on page 190
ZCZGET	Yes, No	DFHZARL2	DFHZARL	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZCZNAC	Yes, No	DFHZARL3	DFHZARL	SUSPEND MISC	User	"SNA LU control waits" on page 190
ZCZNAC	Yes, No	DFHZERH4	DFHZERH	SUSPEND CONV	User	"SNA LU control waits" on page 190
ZXQOWAIT		LIST	DFHZXQO	WAIT_OLDW MISC	System only	"SNA LU control waits" on page 190
ZXQOWAIT		LIST	DFHZXST	WAIT_OLDW MISC	System only	"SNA LU control waits" on page 190

Note:

1. The z/OS Workload Manager (WLM) monitoring environment is set to STATE=IDLE in either of the following situations:
   - A conversational task is waiting for terminal input from its principal facility.
   - A CICS system task is waiting for work.
2. These waits are used by the Front End Programming Interface (FEPI). Problem determination for FEPI is discussed in FEPI error handling.
3. If the task is waiting for resource type ALLOCATE, the current z/OS Workload Manager monitoring environment is set to STATE=WAITING and one of the following conditions is met:
   - RESOURCE=SESS_LOCALMVS, if the session being waited for is a session with another CICS region in the same local MVS image.
   - RESOURCE=SESS_SYSPLEX, if the session being waited for is a session with a CICS region in another z/OS image in the same sysplex.
   - RESOURCE=SESS_NETWORK, if the session being waited for is an ISC session which might be in the same z/OS image.
4. If a terminal is associated with the task.
5. If the task is waiting for resource type IRLINK, the current z/OS Workload Manager monitoring environment is set to STATE=WAITING, RESOURCE=CONV. Look at the RMF workload activity report to see whether the task continued beyond the current WLM monitoring environment. The SWITCHED column in this report can contain the following values:
   - LOCALMVS: the communicating CICS region is on the same local z/OS image.
   - SYSPLEX: the communicating CICS region is on another z/OS image in the same sysplex.
6. "xxx" is literal.
7. The task has not yet started, because the system is at its MAXTASKS (MXT) limit.
8. The task is waiting in the pipeline for another task to complete. These tasks are connected through the Request Stream (RZ) component. The tasks might be using MRO; for example, as part of a Web Services Atomic Transaction that is registering with a coordination service and is waiting for a response. The tasks might be local to the CICS region, in which case MRO is not involved even though the tasks are still using request streams.

9. The task has not yet started because it is being held for transaction class purposes.
10. If the task is waiting for resource type ZCIOWAIT, the current z/OS Workload Manager monitoring environment is set to one of the following states:
    - STATE=IDLE for a conversational task, or DTP transaction, that is awaiting input from its principal facility.
    - STATE=WAITING,RESOURCE=CONV for a task awaiting input from its alternate facility. Look at the RMF workload activity report to see whether the task continued beyond the current WLM monitoring environment. The SWITCHED column in this report can contain the following values:
      - LOCALMVS: the communicating CICS region is on the same local z/OS image.
      - SYSPLEX: the communicating CICS region is on another z/OS image in the same sysplex.
      - NETWORK: the communicating CICS region is in the z/OS Communications Server network, which might be in the same z/OS image.
11. The task is waiting for a request stream request or response from its request stream partner.
12. The task is waiting to send or receive a request or response.
13. The event processing queue server is waiting for an event to be placed on the queue.
14. The event processing dispatcher is waiting for an event to be dispatched.
15. The thread limit for the JVM server has been reached, as specified in the THREADLIMIT attribute of the JVMSERVER resource. The JVM server must wait until a thread becomes available before starting another task. To reduce the frequency of waits for a particular JVM server, increase the value of the THREADLIMIT attribute for that JVMSERVER resource.
16. The event processing deferred filtering task CEPF is waiting for a request to be placed on the queue.
17. The Policy deferred rule evaluation task CMPE is waiting for a request to be placed on the queue.

Asynchronous services waits

This section describes the waits associated with CICS asynchronous services support.

Resource type AS_CHILD

A suspend can occur when a task issues a **FETCH CHILD** command without the NOSUSPEND option specified, and the child task associated with the specified token hasn't completed. The task will resume when the child task completes, or the suspend times out according to the TIMEOUT option on the command.

You can purge the task when it is in this state.

Resource type AS_ANY

A suspend can occur when a task issues a **FETCH ANY** command without the NOSUSPEND option specified, and no unfetched child task is complete. The task will resume when any unfetched child task completes, or the suspend times out according to the TIMEOUT option on the command.

You can purge the task when it is in this state.

Resource type ASPARENT

A parent task can be suspended when it issues a **RUN TRANSID** command if CICS needs to regulate workflow (see Managing performance with the asynchronous API for more information). The task will resume when the workload level in the region drops.

You can purge the task when it is in this state.

Dispatcher waits

The CICS dispatcher might cause tasks to wait, depending on the availability of TCBs in the CICS region and the maximum number of TCBs that CICS is allowed to create for tasks.

The resource names or resource type associated with these dispatcher waits are as follows:

- DS_ASSOC
- DSTSKDEF
- OPENPOOL
- OPEN_DEL
- SSL_POOL
- THR_POOL
- XMCHILD
- XMPARENT
- XP_POOL

Resource type DS_ASSOC

When a task performs an **EXEC CICS LINK** to a Liberty program, or a transaction is attached whose initial program is a Liberty program, CICS will dispatch the task onto a T8 TCB in the Liberty JVM server. The task is suspended on the resource type DS_ASSOC and later resumed on the T8 TCB. If the task is not resumed in the period specified by the **WLP_LINK_TIMEOUT** parameter in the JVM profile, the suspend will timeout. See JVM server options for more information.

Resource type DSTSKDEF

A task waiting on the resource type DSTSKDEF is not suspended. Task attach has added the new task to the dispatcher chain and it is waiting for first dispatch. The task might be waiting for a dump to complete, for example.

Resource name OPENPOOL

CICS automatically sets the limit for the number of open TCBs in the L8 and L9 mode open TCB pool. The limit is based on the maximum number of tasks (MXT or MAXTASKS) specified for the CICS region, using the following formula:

```
(2 * MXT Value) + 32
```

When a task first needs an L8 or L9 mode open TCB, the dispatcher domain tries to find a free TCB of this mode with the correct subspace attributes. If no L8 or L9 mode TCB associated with a matching subspace is free, CICS performs these actions:

- Attaches a new L8 or L9 mode TCB of the required subspace, if the number of open TCBs in the L8 and L9 mode open TCB pool is less than the limit set by CICS, and allocates the new TCB to the requesting task.
- Detaches a free open L8 or L9 mode TCB associated with a different subspace, if there is one available and the limit has been reached, attaches a new L8 or L9 mode TCB, and allocates this new TCB to the requesting task. This process is referred to as TCB stealing: deleting a free TCB of one type to attach one of a different type.

However, if neither of these options is available, the dispatcher places the requesting task onto a queue and the task is suspended, using suspend token AWAITING_OPENPOOL_TOKEN in the DS task block.

When an open TCB becomes free, or if the limit changes, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

Resource name OPEN_DEL

If your task is waiting on a resource name of OPEN_DEL, the dispatcher is detaching an unsuitable TCB (stealing) so that it can allocate a new one, and your task is waiting for the old TCB to end so that the dispatcher can attach a new one.

If your task requires an open TCB, but no suitable TCB is available, and a new TCB cannot be attached because the limit set for the number of open TCBs has been reached, CICS deletes a currently idle TCB to allow the task to attach a TCB of the required type. However, the attach cannot proceed until the process to delete the TCB is complete, otherwise, the number of open TCBs in the pool would temporarily exceed the limit.

Resource name SSL_POOL

When a task first needs a S8 mode open TCB, the dispatcher domain attempts to find a free TCB from the SSL pool. If no S8 mode TCB is free, and the number of open TCBs in the SSL pool is less than the limit specified by the MAXSSLTCBS system initialization parameter, CICS attaches a new TCB and allocates it to the requesting task.

However, if the number of S8 TCBs in the pool is at the limit set by MAXSSLTCBS, dispatcher places the requesting task onto a queue and the task is suspended, using suspend token AWAITING_OPEN_TCB_TOKEN in the DS task block. When an open TCB becomes free, or the MAXSSLTCBS limit is raised, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

Resource name THR_POOL

When a task first needs a T8 mode open TCB, the dispatcher domain attempts to find a free TCB from the THRD pool. If no T8 TCB is free, and the number of open TCBs in the THRD pool is less than the maximum limit, CICS attaches a new TCB and allocates it to the requesting task. The maximum limit for the THRD pool is total number of threads reserved for all the JVM servers in the region, up to a limit of 2000. The number of threads reserved for each JVM server is the THREADLIMIT value on the JVMSERVER resource, plus 1 (the TCB that is reserved for the JVM server).

If the number of T8 TCBs in the pool is at the maximum limit, dispatcher places the requesting task onto a queue and the task is suspended, using suspend token AWAITING_OPEN_TCB_TOKEN in the DS task block. When an open TCB becomes free, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

Resource type XMCHILD

The task is a CICS Business Transaction Services (BTS) child transaction that is suspended waiting for its parent transaction to resume it. The resource name is the task number of the parent transaction.

Resource type XMPARENT

The task is a CICS Business Transaction Services (BTS) parent transaction that is suspended waiting for an associated child transaction to resume it. The resource name is the task number of the child transaction.

Resource name XP_POOL

CICS automatically sets the limit for the number of X8 and X9 TCBs to a value equal to the maximum number of tasks specified for the CICS region (the MXT value).

When a task first needs a X8 or X9 mode open TCB, the dispatcher domain attempts to find a free TCB from the XP pool. If no X8 or X9 mode TCB is free, and the number of open TCBs in the XP pool is less than the limit set by CICS, CICS attaches a new TCB and allocates it to the requesting task.

However, if the number of X8 and X9 TCBs in the pool is at the limit set by CICS, dispatcher places the requesting task onto a queue and the task is suspended, using suspend token AWAITING_OPEN_TCB_TOKEN in the DS task block. When an open TCB becomes free, or if the limit changes, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

CICS DB2 waits

CICS DB2 uses the WAIT_MVS and WAIT_OLDC functions of the CICS dispatcher to put the running CICS task into a wait.

Resource type CDB2CONN

The CICS task has an open TCB but is waiting for a DB2 connection to become available to use with the open TCB. This indicates that the TCBLIMIT value has been reached, which limits the number of open TCBs (and hence connections) that can be used to access DB2. The CICS task must wait for a connection to be freed by another TCB running on behalf of another CICS task, after which it may use the freed DB2 connection with its own TCB.

You cannot purge the task when it is in this state. Message DFHAP0604 is issued at the console if an attempt to forcepurge the task is made. Forcepurge processing is deferred until a DB2 connection has been acquired.

You can increase the number of open TCBs permitted to access DB2 with a SET DB2CONN TCBLIMIT command. If you increase the TCBLIMIT value, CICS posts tasks to retry acquisition of a DB2 connection.

Resource type CDB2RDYQ

The task is waiting for a thread to become available. The resource name details the DB2ENTRY or pool for which there is a shortage of threads.

You cannot purge the task when it is in this state. Message DFHAP0604 is issued at the console if an attempt to forcepurge the task is made. Forcepurge processing is deferred until a thread is acquired.

You can increase the number of threads available for the DB2ENTRY with a SET DB2ENTRY ( ) THREADLIMIT(*nn*) command. You can increase the number of threads available for the pool with a SET DB2CONN THREADLIMIT(*nn*) command. If you increase the THREADLIMIT value, CICS posts tasks to retry acquisition of a thread.

Resource type DB2_INIT

DFHD2IN1 (CICS DB2 initialization program) issues the wait for DFHD2IN2 to complete.

Resource type DB2CDISC

A **SET DB2CONN NOTCONNECTED** command has been issued with the WAIT or FORCE option. DFHD2TM waits for the count of tasks using DB2 to reach zero.

Resource type DB2EDISA

A SET DB2ENTRY DISABLED command has been issued with the WAIT or FORCE option. DFHD2TM waits for the count of tasks using the DB2ENTRY to reach zero.

IBM MQ waits

If a task is waiting on the resource type MQseries, WMQ_INIT, or WMQCDISC, the CICS-IBM MQ adapter has suspended it.

Resource type MQseries

The CICS-IBM MQ MQ adapter (DFHMQTRU module) put the task into a CICS wait because the WAIT option was used with the MQGET call and there was no message available. The resource name used for the wait is GETWAIT. The WAIT_MVS function of the dispatcher is used for this wait, and the wait type for workload management is OTHER_PRODUCT. The task can be purged.

Resource type WMQ_INIT

DFHMQIN1, the CICS-IBM MQ initialization program, issues this wait for DFHMQIN2 to complete. The WAIT_OLDC function of the dispatcher is used for this wait, and the wait type for workload management is MISC. The task can be purged.

Resource type WMQCDISC

A SET MQCONN NOTCONNECTED command has been issued with the WAIT or FORCE option, and the DFHMQTM module waits for the count of user tasks using IBM MQ to reach zero. The resource name is given as the name of the installed MQCONN resource definition for the CICS system. The WAIT_OLDC function of the dispatcher is used for this wait, and the wait type for workload management is MISC. The task can be purged.

DBCTL waits

DBCTL can enter the wait state for three reasons: the connection to DBCTL using CICS-supplied transaction CDBC has failed to complete, a user task is waiting on the DBCTL resource DLSUSPND, or the attempt to disconnect from DBCTL using CICS-supplied transaction CDBC has failed to complete.

Connection to DBCTL has failed to complete

Connection to DBCTL using the CICS-supplied transaction CDBC takes place in two phases. You can find the current phase using either transaction CDBC, by refreshing the screen display, or transaction CDBI.

In phase 1, CDBC passes the request for connection to IMS, and returns. It is very unlikely for a wait to occur during this phase, unless there is an error in CICS code. In such a case, you would see this message displayed whenever you inquired on the connection status using CDBI:

```
DFHDB8291I DBCTL connect phase 1 in progress.
```

In phase 2, IMS processes the request asynchronously, and returns to CICS when connection is complete. Until the connection is complete, you see this status message displayed whenever you inquire with CDBI:

```
DFHDB8292I DBCTL connect phase 2 in progress.
```

If this phase fails to complete, the failure is associated with IMS. See Diagnosis in IMS product documentation for guidance about debugging the problem.

A user task is waiting on resource type DBCTL

If you find that a user task is waiting on a resource type of DBCTL and resource name DLSUSPND, the task has made a DL/I request. The task is suspended by CICS while the request is serviced by DBCTL. If the task has not resumed, the request has not completed.

Disconnection from DBCTL has failed to complete

When you use CDBC to disconnect from DBCTL, it invokes another CICS transaction, CDBT. CDBT makes the disconnection request to DBCTL and is suspended by CICS while DBCTL services the request asynchronously.

If disconnection fails to complete, you can inquire on CDBT using, for example, CEMT INQ TASK to see how far disconnection has progressed. You will probably find that CDBT is waiting on resource type DBCTL and resource name DLSUSPND, in which case the request is being processed by DBCTL.

- If CDBT *is* waiting on DBCTL, what you do next depends on whether you have requested “orderly” or “immediate” disconnection.
  - If you have requested “orderly” disconnection, it is likely that DBCTL is waiting for conversational tasks to finish. You can override an “orderly” disconnection by requesting “immediate” disconnection, in which case the process should end at once.
  - If you have requested “immediate” disconnection, and this does not happen, there is an unexpected wait within IMS. See Diagnosis in IMS product documentation for guidance about investigating the problem.
- If CDBT *is not* waiting on DBCTL, this indicates a problem with CICS code. Contact the IBM Support Center for further assistance.

EDF waits

A user task is made to wait on resource type EDF and resource name DBUGUSER when, under the EDF session, CICS has control for EDF processing.

Log manager waits

Read this section if the resource type your task is waiting on starts with the characters LG, indicating log manager.

The journal name, given as the resource name, refers to the last element of the MVS log stream name. For example, in the log stream name PAYRO.ACC0001.UJ4321, the journal name, for these purposes, is UJ4321. If you do encounter any of these waits, look at the MVS console for messages prefixed with ‘IXG'. These are the MVS system logger messages and might provide further information about the cause of the wait. The MVS system console might also reveal evidence of resource contention within MVS, a possible cause of a log manager wait.

If the task is writing to a journal on an SMF log, the journal name is the name of the journal.

Do not use the **SET TASK** command to remove tasks that are waiting for the log manager resource types. If you use this command to purge the task, CICS ignores the command. If you use the FORCEPURGE or KILL options, the CICS region might fail.

Resource type LG_DEFER
The task is the first task to request that the currently active log buffer be flushed. The task waits for 30 milliseconds to allow other tasks to append more records to the buffer.

Resource type LG_FORCE
The task is waiting for the flush of a log buffer to complete. It is resumed by the task that performs the flush operation. The task can be purged if the log stream is not DFHLOG, the primary system log.

Resource type LG_RETRY
This is a temporary error, reading or writing a log from a log stream. CICS waits and retries the operation.

Resource type LGDELALL
During an initial start of CICS, CICS calls the MVS system logger macro IXGDELET ALL. CICS waits until the MVS system logger posts the ECB.

Resource type LGDELRAN
During keypoint processing, CICS calls the MVS system logger macro IXGDELET RANGE. CICS waits until the MVS system logger posts the ECB.

Resource type LGENDBLK
During an emergency restart of CICS, or transaction backout, CICS calls the MVS system logger macro IXGBRWSE END. CICS waits until the MVS system logger posts the ECB.

Resource type LGENDCRS
During an emergency restart of CICS, CICS calls the MVS system logger macro IXGBRWSE END. CICS waits until the MVS system logger posts the ECB.

Resource type LGFREVER
When CICS is quiescing and a dynamic backout fails because a task fails to read the system log, the task is suspended 'forever' to allow other tasks to continue to back out. The SDTRAN process deletes these suspended tasks during CICS shutdown.

Resource type LGHARTBT
The log manager 'heartbeat' system task checks that the MVS Logger connection to the system log is still valid. The task is in this wait state most of the time. There must be one task in this state on the system.

Resource type LGREDBLK
During an emergency restart of CICS, or transaction backout, CICS calls the MVS system logger macro IXGBRWSE READBLOCK. CICS waits until the MVS system logger posts the ECB.

Resource type LGREDCRS
During an emergency restart of CICS, CICS calls the MVS system logger macro IXGBRWSE READCURSOR. CICS waits until the MVS system logger posts the ECB.

Resource type LGSTRBLK
During an emergency restart of CICS, or transaction backout, CICS calls the MVS system logger macro IXGBRWSE START. CICS waits until the MVS system logger posts the ECB.

Resource type LGSTRCRS
During an emergency restart of CICS, CICS calls the MVS system logger macro IXGBRWSE START. CICS waits until the MVS system logger posts the ECB.

Resource type LGWRITE
In several situations, CICS calls the MVS system logger macro IXGWRITE. CICS waits until the MVS system logger posts the ECB.

RRMS waits

If a task is waiting on the resource type RRMSEXIT, the RX domain has suspended the task.

GET_CLIENT_INFO
The RX domain waits on the next inbound request from the external client for the unit of recovery.

NOTIFICATION
The system task waits on an ECB. When an ECB is posted, the task checks the current state of the resource manager.

RESYNC
The system task waits on an ECB. When an ECB is posted, the task executes resync for any units of recovery on the resync chain.

SYNCPOINT
The RX domain waits for the RRS to deliver the overall decision for the unit of recovery.

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.

USERWAIT indicates that a task has issued an **EXEC CICS WAIT EVENT EXTERNAL** or an **EXEC CICS WAITCICS** command.

EKCWAIT indicates that a task has issued an **EXEC CICS WAIT EVENT** command.

If the wait is prolonged, you should identify the event being waited on, and:

- Check that the **EXEC CICS WAIT EVENT** 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 and the **EXEC CICS WAIT EVENT** command included the NAME option, the specified name is the resource name. For programming information about the NAME option of the **WAIT EVENT** command, see WAIT EVENT.

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 17 on page 189.

Table 17. 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 “Investigating terminal waits” on page 118.)

Check the contents of the SUSPAREA entry. Does it contain a valid address? That is, is it within the CICS address space, and 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 “Dealing with storage violations” on page 231 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 - wait for file state changes" on page 144.
- 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 ENQUEUE - waits for locks on files or data tables" on page 150.

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 or a DFHKC TYPE=DEQ macro call. 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.

SNA LU control waits

Systems Network Architecture (SNA) logical unit (LU) control waits are associated with the following resource types. The implication of waits on any of these resource types are also described.

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 RTIMOUT value of the profile used by the transaction. 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

DFHZSLS has to set the TCT prefix SNA fields from the ACB. This wait is issued to ensure that these fields are set before being used.

Resource type ZC_ZGCH

DFHZGCH is waiting for the SNA CHANGE ENDAFFIN macro to complete.

Resource type ZC_ZGIN

DFHZGIN issues the SNA INQUIRE macro and waits until SNA completes execution of this request.

Resource type ZC_ZGRP

- If the task is waiting on resource name PSINQECB, this means that DFHZGRP has issued the SNA macro INQUIRE PERSESS during SNA persistent session restart, or during the reopening of the SNA ACB, and is waiting for a response from SNA. The wait expires after 5 minutes if SNA does not respond.
- If the task is waiting for resource name PSOP2ECB, this means that DFHZGRP has issued the SNA macro OPNDST RESTORE during emergency restart, and is waiting for a response from SNA. The wait expires after 5 minutes if SNA does not respond.

Resource type ZC_ZGUB

DFHZGUB issues ten SNA CLSDST or TERMSESS macros during persistent sessions restart. It waits for an RPL to become free for SNA to post the SNA exit. The wait expires after 5 minutes if SNA 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 LU6.2 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 Effects of the MAXIMUM option of the SESSIONS resource.

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 "Investigating terminal waits" on page 118.

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 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 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 described in the following information. Note that 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 that is being run 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 yet be serviced, the program 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 162), 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 that corresponds 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 the following reasons:

- A task needs to change TCB 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 QSAM to write data to extrapartition transient data destinations. QSAM runs synchronously with tasks that are requesting its services. This means that any task that invokes a QSAM service must wait until the QSAM processing is complete. If QSAM enters an extended wait for any reason, the requesting task also experiences an extended wait.

  An extended wait might occur when QSAM attempts to access an extrapartition data set. QSAM uses the MVS RESERVE volume-locking mechanism to gain exclusive control of volumes while it accesses them, which means that any other region that attempts to write to the same volume is forced to 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.

Resource types TDIPLOCK, ENQUEUE, RRMSEXIT, TD_READ, Any_MBCB, Any_MRCB, MBCB_xxx, and MRCB_xxx

If your task is waiting on any of the resource types TDIPLOCK, ENQUEUE, RRMSEXIT, TD_READ, 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: waits for transient data intrapartition requests

If you have a task suspended on resource type TDIPLOCK, with a resource name that corresponds 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.

For more information about the constraints that apply to tasks writing to intrapartition destinations, see Transient data control.

Resource type ENQUEUE

If a transient data queue has been defined as intrapartition and logically recoverable, there are further restrictions on the use of the queue by more than one task at a time (in addition to those leading to waits on resource type TDIPLOCK).

If you have a task suspended on resource type ENQUEUE, and a value of TDNQ, the task has been suspended while attempting to read, write or delete a logically recoverable queue because a required enqueue is currently held by another task.

Note: For general information about dealing with enqueue waits, see "Investigating enqueue waits" on page 131. Issuing a **CEMT INQUIRE UOWENQ** command reveals the name of the queue and whether the enqueued read or write is required by the task. If the task is enqueued against the read end of the queue, a qualifier of FROMQ is displayed on the **CEMT INQUIRE UOWENQ** screen. If the task is enqueued against the write end of the queue, a qualifier of TOQ is displayed on the **CEMT INQUIRE UOWENQ** screen.

If you want to delete a queue, both the read and the write enqueues must be obtained. No task may, therefore, read or write to a queue while a delete operation is in progress. A delete cannot proceed until any task currently reading has completed its read or any task writing has committed its changes.

In general, a wait on a resource type of ENQUEUE should not last for long unless the task owning the enqueue has been delayed. If the UOW that owns the enqueue has suffered an indoubt failure, the UOW is shunted. If the queue accessed by this UOW is defined as WAIT=YES and WAITACTION=QUEUE, the wait can last for a long period of time. To deduce if an indoubt failure has occurred:

- Issue a **CEMT INQUIRE UOWENQ** command to display the name of the enqueue owner.
- Issue a **CEMT INQUIRE UOW** command to see if the UOW is shunted.

Resource type RRMSEXIT

Resource type TD_READ

If a queue is defined as logically recoverable, a TD_READ wait may be encountered.

A task can read from a queue while another task is writing to the same queue. If this happens, the first task holds the read enqueue and the second task holds the write enqueue on the queue. The task reading the queue can only read data that has already been committed. It cannot read data that is currently being written to the queue until the task holding the write enqueue commits the changes it has made and dequeues from the write end of the queue.

A task is suspended on a resource type of TD_READ if it is trying to read uncommitted data from a logically recoverable queue. The queue name is displayed in a qualifier. The suspended task is forced to wait until the task owning the write enqueue commits the changes it has made..

In most cases, the suspended task will not have to wait long. A lengthy wait can occur if the task owning the write enqueue suffers from an indoubt failure (which causes the associated UOW to be shunted), and the queue is defined with the WAIT=YES and WAITACTION=QUEUE attributes.

If you do not want to wait for data to be committed to the queue, code NOSUSPEND on the READQ TD request. QBUSY is returned to the application and the task does not wait.

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” on page 194 and “Resource type MRCB_xxx” on page 194).

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.

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 INQ 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 that shows there is a system error, contact your IBM Support Center.

FEPI waits

This section outlines the CICS waits that FEPI issues.

Table 18 on page 195 shows the points at which FEPI issues CICS waits:

Table 18. FEPI waits

Resource name	Resource type	Wait type	Description
FEPI_RQE	ADAPTER	WAIT_MVS	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_MVS	Issued in the FEPI Resource Manager when it has no work to do. Ends when work arrives (from either the FEPI adapter or a z/OS Communications Server 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 TCB 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.

Recovery manager waits

This section describes waits associated with the CICS recovery manager (RM).

Resource type RMCLIENT

If a task is suspended with a resource type of RMCLIENT, the recovery manager is trying to call a client which has not yet registered or set its gate. Clients register with the recovery manager and set their gates during CICS initialization, so the suspended task should be resumed by the time CICS initialization is complete.

If such a task does remain suspended for a long time after CICS initialization completes, there is probably an error in CICS. Contact your IBM Support Center.

Resource type RMUOWOBJ

- If a task is suspended with a resource type of RMUOWOBJ and a resource name of LOGMOVE, the recovery manager is trying to log data for a unit of work while an activity keypoint which is moving the UOW's log data is in progress. The suspended task should be resumed when the activity keypoint task completes the move of the UOW's log data.

  If a task remains suspended for a long time with a resource type of RMUOWOBJ and a resource name of LOGMOVE, try to discover why the activity keypoint task (CSKP) is not completing.
- If a task is suspended with a resource type of RMUOWOBJ and a resource name of EXISTENC, the recovery manager is trying to delete a unit of work while an activity keypoint is in progress. The suspended task should be resumed when the activity keypoint task finishes working with the UOW.

  If a task remains suspended for a long time with a resource type of RMUOWOBJ and a resource name of EXISTENC, try to discover why the activity keypoint task (CSKP) is not completing.

CICS Web waits

This section describes waits associated with CICS web support.

Resource type WBALIAS

A suspend can occur on the CICS WEB attach transaction after it has attached its partner (WEB alias) transaction. This suspend only occurs if the client socket is using SSL to communicate with CICS. The suspend is resumed when the WEB alias transaction terminates.

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 CICS might experience symptoms such as high CPU usage and transaction abends.

The list of symptoms are described in "Loops" on page 12. 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 section outlines procedures for finding which programs are involved in a loop that does not terminate. It contains the following topics:

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. In turn, the loops that CICS can detect can be classified into tight loops and non-yielding loops.

Figure 30 on page 197 gives an example of code containing a simple tight loop.

```
   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 30. 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.

For information about how CICS handles a looping or runaway task that runs in a JVM server, see CICS task and thread management.

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 region 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. In the extreme case, there could be a single instruction in the loop, causing a branch to itself.

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 neither suspend the program nor pass control to the dispatcher. 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 intended record is already in a VSAM buffer. There is, therefore, no point at which the task can be suspended, and so the ICVR cannot be reset.

Figure 31 on page 198 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 31. 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 199 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 162. 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 32 on page 199 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 32. 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 12.

If you suspect that you have a yielding loop, turn to "Investigating loops that are not detected by CICS" on page 202 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.

About this task

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.

Procedure

1. Get the documentation you need.
2. Look at the evidence.
3. Identify the loop, using information from the trace table and transaction dump.
4. Determine the reason for the loop.

Results

Use the following information to complete the steps above.

Getting the documentation you need

Before you begin

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.

About this task

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.

Procedure

1. You need to 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:
   a) Select level-1 special tracing for AP domain, and for the EXEC interface program (EI).
   b) Select special tracing for just the task that has the loop, and disable tracing for all other tasks by turning the main system trace flag off.

   You can find guidance about setting up these tracing options in Using CICS trace.
2. Start the task, and wait until it abends AICA.
3. 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 39.)

Results

You now have the documentation you need to find the loop.

Looking at the evidence

Once you have collected the necessary documentation, use the following guidance to analyze the information you have gathered.

Procedure

1. 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**”.
2. 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.

Results

If you find that the loop is within CICS code, you need to 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

To identify loops in user programs, you can look in the transaction dump or you can use the trace table.

Procedure

- To identify a loop by using the transaction dump, use the following steps:
  a) Find the program status word (PSW), and see whether 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.

  b) Use the module index at the end of the formatted dump to find the module name of the next instruction.

     If the instruction address is not in your code, it is less useful for locating the loop. However, try to identify the module that contains the instruction, because it is probably the one that was called during the execution of a CICS request made within the loop. 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 is in a module owned by CICS or another product, and your program is not responsible for it. If the loop is in CICS code, contact the IBM Support Center.

  c) If the PSW points 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.

  d) When you have located a point within the loop, work through the source code and try to find the limits of the loop.
- To identify a loop by using the trace table, use the following steps:
  a) Go to the last entry in the internal trace table, and work backwards 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.

  b) Make a note of the task number, so you can check that any other trace entries you read relate to the same abended task.

  c) Look at the entries preceding AP 1942. In particular, look for trace entries with the point ID AP 00E1.

     These entries 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). Entries with a point ID of AP 00E1 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.

  d) When you have identified the value of EIBFN, use the function code list in Function codes of EXEC CICS commands to identify the command that was issued.

  e) 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.

Finding the reason for the loop

When you have identified the limits of the loop, you need to 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 or GTF 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, see some sort of repetitive output, or statistics might show an excessive number of I/O operations or requests for storage. These types of symptom can indicate that you have a yielding loop.

About this task

The nature of the symptoms might 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.

Procedure

1. 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 Using CICS trace.

   a) Select level-1 special tracing for every component. You need to capture as much trace information for the task as possible, because you do not yet know what functions are involved in the loop.

   b) Set all standard tracing off, by setting the main system trace flag off.

   c) Select special tracing for just the task containing the loop.

   d) 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.

2. When you have captured the trace data, you need to purge the looping task from the system.

   a) Use the **CEMT INQ TASK** command to find the number of the task.

   b) Purge the task using either the **CEMT SET TASK PURGE** or the **CEMT SET TASK FORCEPURGE** command.

      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.

   This causes the transaction to abend, and to produce a transaction dump of the task storage areas.

3. In addition to the auxiliary trace data and the transaction dump, get the source listings of all the programs in the transaction.

Results

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

About this task

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.

Procedure

1. 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.
2. If you are able to detect a pattern, you should be able to identify the corresponding pattern of statements in your source code.

Finding the reason for the loop

Before you begin

About this task

Procedure

1. Look carefully at the statements contained in the loop. Does the logic of the code suggest why the loop occurred?
2. 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.

Example

What to do next

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.

About this task

Procedure

1. Use the interactive tools that CICS provides.

- 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.
- Use CECI and CEBR 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.

2. Modify the program, and execute it again.

   If the program is extremely complex, or the data path difficult to follow, you might need to insert additional statements into the source code.

   - Adding extra ASKTIME commands allow you to use EDF and inspect the program at more points.
   - Request dumps from within your program, and insert user trace entries, to help you find the reason for the loop.

Dealing with performance problems

When you have a performance problem, you might be able to find that it is 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 "Why tasks fail to get attached to the transaction manager" on page 205.
2. Some tasks fail to get attached to the dispatcher—see "Why tasks fail to get attached to the dispatcher" on page 206.
3. Some tasks get attached to the dispatcher, but fail to get dispatched—see "Why tasks fail to get an initial dispatch" on page 208.
4. Tasks get attached to the dispatcher and then run and complete, but take a long time to do so—see "Why tasks take a long time to complete" on page 209.

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 204.

There is a quick reference section at the end of this section ("A summary of performance bottlenecks, symptoms, and causes" on page 210) that summarizes bottlenecks, symptoms, and actions that you should take.

Finding the bottleneck

Four potential bottlenecks can be identified for user tasks, and three for CICS system tasks.

About this task

The bottlenecks are:

- Attach to transaction manager (user tasks only)
- Attach to dispatcher (user tasks and system tasks)
- Initial dispatch (user tasks and system tasks)
- Dispatch, suspend and resume cycle (user tasks and system tasks)

Procedure

1. Determine which bottleneck is causing your performance problem.

   Each bottleneck is affected by a different set of system parameters and you might find that adjusting the parameters solves the problem.

2. If performance is particularly poor for any of the tasks in your system, you might be able to capture useful information about them with the command **`CEMT INQ TASK`**.

However, tasks usually run more quickly than you can inquire on them, even though there might be a performance problem. You can use performance class monitoring or tracing to get the information you require.

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 INQ 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 "Why tasks fail to get attached to the transaction manager" on page 205.

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 INQ TASK** shows 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 INQ 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 INQ 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 "MXT summary" on page 157.

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 INQ TASK** returns a status of 'Dispatchable' 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 INQ 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 "Why tasks fail to get an initial dispatch" on page 208.

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 "Why tasks take a long time to complete" on page 209.

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 "Investigating interval control waits" on page 64. You need to 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. You can check its status using **CEMT INQ 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.

   You can use **CEMT INQ TERMINAL** to inquire on the status of the remote system.

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 that the system has reached the maximum number of tasks (MXT) limit or the task belongs to a transaction class that has reached its MAXACTIVE limit.

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.

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.

You might be able to use **CEMT INQ TASK** to show that 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. For more information on setting MXT, see Setting the maximum task specification (MXT) .

Using transaction manager statistics

You can use the transaction global statistics and transaction class statistics to see whether the MXT and transaction class limits are adversely affecting performance.

About this task

To find out how often the MXT and transaction class limits are reached, look at the transaction global statistics and transaction class statistics. You can compare the number of times these limits are reached with the total number of transactions and see whether the values set for the limits are adversely affecting performance.

Procedure

1. 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 SMF data set.
2. Format this data set by using the statistics utility program, DFHSTUP.

   You might find the following DFHSTUP control parameters useful:

   ```
   SELECT APPLID=
   COLLECTION TYPE=
   REQTIME START= ,STOP=
   DATE START= ,STOP=
   ```

If you correctly code these control parameters, you avoid formatting information that is unnecessary at this point. For information about the DFHSTUP utility, see Statistics utility program (DFHSTUP).

Results

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.

What to do next

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 CICS monitoring facility: Performance and tuning .

Using CICS monitoring

You can use monitoring information to find out how long an individual task waits to be attached to the dispatcher.

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 Monitoring.

Using trace

You can use trace if you want to find out just how long an individual task waits to be attached to the dispatcher.

About this task

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.

Procedure

1. Select special tracing for the transaction associated with the task, and turn off all standard tracing by setting the main system trace flag off.
2. 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 Using CICS trace.
3. When you have selected the options, start tracing to the internal trace table and attempt to initiate the task.
4. When the task starts, get a system dump using the command **CEMT PERFORM SNAP**. Format the dump using the keyword TR, to get the internal trace table.
5. 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.

What to do next

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, you need to 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 the value you calculate is significantly greater than 0, the dispatcher could not dispatch the task immediately.

The following factors influence the length of time that a task must wait before getting its first dispatch:

- The priority of the task
- Whether the system is becoming short on storage
- Whether the system is 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 cannot control 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.

How storage conditions impact new tasks

CICS attempts to alleviate storage stress conditions by releasing programs with no current user, and by not attaching new tasks.

If these actions fail to eliminate storage stress, or if the short-on-storage (SOS) condition is caused by a suspended GETMAIN, one or more of the following 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)
DFHSM0606 applid
 The amount of MVS above the bar storage available to CICS is critically low
```

If you do not observe the SOS messages, you can find out how many times CICS became SOS from the storage manager statistics (the "Times went short on storage" statistic). You can also get this information from the storage manager domain DSA summary in a formatted system dump.

For more information about short-on-storage conditions, see Short-on-storage conditions in dynamic storage areas.

The dispatcher recognizes two other conditions on the approach to an SOS condition:

- storage getting short

- storage critical

These two conditions affect the chance of new tasks getting a first dispatch. From the point when storage gets short, through to when storage gets critical and up to the SOS condition, the priorities of new user tasks are reduced in proportion to the severity of the condition. However, this is not true if the **PRTYAGE** system initialization parameter is set to 0. At first, you are not likely to notice the effect, but as the "storage critical" condition 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" conditions occur many times for every occasion that the SOS condition is reached. 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.

The following table summarizes the effects of storage conditions on task priorities:

Table 19. 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

Why tasks take a long time to complete

The purpose of this section is to deal not with waiting tasks, but instead with tasks that complete more slowly than they should.

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.

Here are some factors that can affect how long tasks take to complete.

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 Improving the performance of a CICS system. 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 timeout interval on performance

The timeout 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 timeout interval is likely to hold on to resources longer than a task with a short timeout interval. Such a task has a greater chance of preventing other tasks from running. It follows that task timeout intervals should be chosen with care, to optimize the use of resources by all the tasks that need them.

The distribution of data sets on DASD volumes

CICS uses QSAM to write data to extrapartition transient data destinations, and QSAM uses the MVS RESERVE mechanism. If the destination happens to be a DASD volume, any other CICS regions trying to access data sets on the same volume are held up until the TD WRITE is complete.

Other system programs also use the MVS RESERVE mechanism to gain exclusive control of DASD volumes, making the data sets on those volumes inaccessible to other regions.

If you notice in particular that tasks making many file accesses take a long time to complete, check the distribution of the data sets between DASD volumes to see if volume locking could be the cause of the problem.

A summary of performance bottlenecks, symptoms, and causes

This summary includes the symptoms you get if the performance of your system is restricted, and the specific causes of the delays at each point.

Table 20. A summary of performance bottlenecks, symptoms and causes

Bottleneck	Symptoms	Possible causes
Initial attach to transaction manager	**CEMT INQ 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 INQ 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 short-on-storage (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 timeout interval too large • CICS data sets are on volumes susceptible to MVS RESERVE locking

Dealing with incorrect output

Incorrect output has been categorized into a number of areas. Look at the appropriate section to diagnose why this problem is occurring.

The various categories of incorrect output are dealt with in:

- “Trace output is incorrect” on page 211
- “Dump output is incorrect” on page 214
- “Incorrect data is displayed on a terminal” on page 217
- “Specific types of incorrect output for terminals” on page 218
- “Incorrect data is present on a VSAM data set” on page 222
- “An application does not work as expected” on page 223
- “Your transaction produces no output at all” on page 223
- “Your transaction produces some output, but it is wrong” on page 229.

Trace output is incorrect

If you have been 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 to make sure you get the tracing you want.

There are two main types of problem:

- Your tracing might have gone to the wrong destination. This is dealt with in "Tracing has gone to the wrong destination" on page 211.
- You might have captured the wrong data. This is dealt with in "You have captured the wrong trace data" on page 212.

Tracing has gone to the wrong destination

In terms of destinations, CICS system trace entries belong to one of three groups - internal tracing, auxiliary tracing and GTF tracing.

- CICS trace entries, other than CICS z/OS Communications Server (SNA) exit traces and exception traces, go to any of the following trace destinations that are currently active:
  - The internal trace table
  - The current auxiliary trace data set
  - The GTF trace data set.
- CICS Communications Server exit traces that are not exception traces go only to the GTF trace data set, if GTF tracing is active.
- CICS Communications Server exit traces that are exception traces go to the internal trace table and, if GTF tracing is active, to the GTF trace data set.
- All other CICS exception traces go to the internal trace table and to any other trace destination that is currently active.

For CICS system tracing other than exception traces and CICS Communications Server exit traces, you can inquire on the current destinations and set them to what you want using the CETR transaction.

CETR - trace control 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 GTF trace destination. If an exceptional condition occurred, the corresponding exception trace entry would be made both to the GTF data set and to the internal trace table, even though the internal trace status is STOPPED.

Note that the main system trace flag value only determines whether standard tracing is to be done for a task (see Table 8 on page 73). 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 no matter what the 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.

GTF tracing
goes to the GTF trace data set. GTF tracing must be started under MVS, using the TRACE=USR option, before the trace entry can be written. Note that if GTF tracing has not been started in this way, the GTF tracing status can be shown as STARTED on the CETR screen and yet no trace entries are made, and no error condition reported.

You have captured the wrong trace data

There are several ways in which you might capture the wrong trace data. The following points 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 certain 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 212 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 212 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 213 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 CICS 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 Table 7 on page 72.

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.

1. 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.
2. 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.

   Table 8 on page 73 illustrates the logic used to determine whether a trace call is to be made from a trace point.

3. 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" on page 212.

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.

These cases are considered:

- The trace data you wanted did not appear at the expected time.
- The earliest trace entry in the table was time-stamped after the activity that interested you took place.
- You could not find the exception trace entry you were expecting.

If the trace entry did not appear at the expected time, consider these possibilities:

- 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, see "You are not getting the correct task tracing" on page 212 and "You are not getting the correct component tracing" on page 212.
- If you were using GTF tracing, it might not have been active at the time the trace entry should have been made. GTF tracing must be started under MVS using the TRACE=USR option.
- If CICS z/OS Communications Server exit trace entries (point IDs AP FCxx) were missing, remember that they are only ever made to the GTF trace data set.
- 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 â€œtransaction attachâ€ (point ID XM 1102) trace entry. This could occur if the main system trace flag is switched off and the transaction status is set to special, or if you do not have KC level 1 tracing selected.

  When you select trace entries by specifying **TRANID** or **TERMID** parameters in the DFHTU710 trace control statements, DFHTU710 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 DFHTU710, see Trace utility print program (DFHTU700).

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.

If the earliest trace entry was later than the event that interested you, and 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 or GTF trace destinations.

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.

GTF trace always wraps when the data set is full. If this was your trace destination, try increasing the size of the GTF trace data set.

If 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 do not get the dump output you expect.

The things that can go wrong are:

- The dump does not seem to relate to you CICS region.
- You do not get a dump when an abend occurs.
- Some dump IDs are missing from the sequence of dumps in the dump data set.
- You do not get the correct data when you format a system dump.

The sections that follow give guidance about resolving each of these problems in turn.

The dump does not seem to relate to your CICS region

If you have experienced this problem, it is likely that you have dumped the wrong CICS region. It should not occur if you are running a single region.

If you invoked the dump from the MVS console using the MVS MODIFY command, check that you specified the correct job name. It must be the job used to start the CICS region in which you are interested.

If you invoked the dump from the CICS main terminal using **CEMT PERFORM SNAP**, check that you were using the main terminal for the correct region. This is more likely to be a problem if you have an SNA network, because that allows you to switch a single physical SNA LU between the different CICS regions.

You do not get a dump when an abend occurs

Read this section if you are experiencing any of these problems:

- A transaction abended, but you do not get a transaction dump.
- A transaction abended and you get a transaction dump, but you do not get the system dump you want at the same time.
- A system abend occurred, but you do not get a system dump.

There are, in general, two reasons why dumps might not be taken:

- Dumping is suppressed because of the way the dumping requirements for the CICS region 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.
  - The system dump data set was being written to by another region, and the DURETRY time was exceeded.
  - 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. Use the CMAC transaction or see CICS messages for guidance about the action to take.

How dumping can be suppressed

If you do 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:

- System dumps can be globally suppressed.
- System dumps and transaction dumps can be suppressed for specific transactions.
- System dumps can be suppressed for specific dump codes from a dump domain global user exit program.
- System dumps and transaction dumps can be suppressed by dump table options.

You need to find out which of these types of dump suppression apply 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 a value of NO for the DUMP parameter in the system initialization table.
- 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 either of these means, any system dumping requirements specified in the transaction dump table and the system dump table are overridden.

You can inquire whether system dumping has been suppressed globally by using the **EXEC CICS INQUIRE SYSTEM DUMPING** system programming command. If necessary, you can cancel the global suppression of system dumping using **EXEC CICS SET SYSTEM DUMPING** with a CVDA value of 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 Global user exit programs.

If an exit program that suppresses system dumping for a particular dump code is enabled, system dumping is not done 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, you can 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

Transaction dumps taken when a transaction abends can be suppressed for individual transactions by using the **EXEC CICS SET TRANSACTION DUMPING** system programming command, or by using the DUMP attribute on the RDO definition of the transaction. None of the dumping requirements specified in the transaction dump table would be met if a transaction for which dumping is suppressed were to abend.

You can 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.

You can inquire on transaction and system dump code attributes using **CEMT INQ TRDUMPCODE** and **CEMT INQ SYDUMPCODE**, respectively. You must specify the dump code you are inquiring on.

If you find that the dumping options are not what you want, you can use **CEMT SET TRDUMPCODE** code or **CEMT SET SYDUMPCODE** code to change the values of the attributes accordingly.

- **If you had no transaction dump when a transaction abended**, look first to see if attribute 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 INQ TRDUMPCODE** and check whether there is an attribute of SYSDUMP or NOSYSDUMP for the dump code. You need to have 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 INQ 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 are 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.

Table 21 on page 216 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.

Table 21. Typical distribution of dump IDs between dump data sets

On system dump data set	On transaction dump data set
ID=23/0001	
ID=23/0002	ID=23/0002
	ID=23/0003
ID=23/0004	
	ID=23/0005
ID=23/0006	

Table 21. Typical distribution of dump IDs between dump data sets (continued)	
On system dump data set	**On transaction dump data set**
ID=23/0007	
	ID=23/0008

For further discussion of the way CICS manages transaction and system dumps, see "Using dumps in problem determination" on page 18.

You do not get the correct data when formatting the 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 specified at the MVS level, for example "without LPA". This requirement would be recorded in the MVS parameter library.

Incorrect data is 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, batch regions, IMS regions and so on, 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:

1. The identity of the transaction associated with the incorrect output.
2. For an autoinstalled terminal, the model number, to ensure that you inquire on the correct TERMTYPE. 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 PROFILE definitions for the transaction, and the TYPETERM definitions for the affected terminal. The attributes most likely to be of interest are SCRNSIZE for the PROFILE, and ALTSCREEN, ALTPAGE, PAGESIZE, EXTENDEDDS, and QUERY for 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.

Tools for debugging terminal output in a z/OS Communications Server environment

Among the debugging tools you have, two are likely to be of particular use for investigating terminal incorrect output errors in a z/OS Communications Server environment.

These debugging tools are as follows:

z/OS Communications Server buffer trace
This is a function of the Communications Server. You need to read the appropriate manual in the z/OS Communications Server library to find out how to use it.

CICS Communications Server exit trace
This is a function of CICS, and you can control it from the CETR panel.

For information on using tracing in CICS problem determination, see Using CICS trace.

Specific types of incorrect output for terminals

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 TYPETERM 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 TYPETERM definitions that you supply, but YES is the value for TYPETERM definitions that are supplied with CICS.

If you have a persistent problem with logon rejection, you can use the z/OS Communications Server 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. You can look up messages in CICS messages. 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 a PROFILE, and too many rows have been specified for ALTSCREEN and ALTPAGE in the TYPETERM 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 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 upper and 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 must look at the options governing the translation.

These are the significant properties of the various translation options you have:

- The ASIS option for BMS or terminal control specifies that lowercase characters in an input data stream are *not* to be translated to uppercase.

  ASIS overrides the UCTRAN attributes for both TYPETERM and PROFILE definitions.
- The UCTRAN attribute of the TYPETERM definition states whether lowercase characters in input data streams are to be translated to uppercase for terminals with this TYPETERM definition.

  The UCTRAN attribute of TYPETERM is overridden by ASIS, but it overrides the UCTRAN attribute of a PROFILE definition.
- The UCTRAN attribute of a PROFILE states whether lowercase characters in the input data stream are to be translated to uppercase for transactions with this PROFILE running on SNA logical units (LUs). The PROFILE UCTRAN value is valid only for SNA LUs.

  The UCTRAN option for a PROFILE is overridden by both the UCTRAN option for a TYPETERM definition and the BMS or terminal control ASIS option.
- If the ASIS option is NOT specified, then if either the PROFILE or the TYPETERM definitions specify UCTRAN(YES), the data presented to the transaction IS translated.

Note: You can also use the user exit XZCIN to perform uppercase translation.

The UPPERCASE option in the offline utilities (DFHSTUP, DFHDU710, DFHTU710) specify whether all lowercase characters are to be translated to uppercase characters.

Table 22 on page 219 and Table 23 on page 219 summarize whether you get uppercase translation, depending on the values of these options.

Table 22. Uppercase translation truth table — ASIS option not specified

Profile	TYPETERM UCTRAN(YES)	TYPETERM UCTRAN(NO)
UCTRAN(YES)	Yes	Yes
UCTRAN(NO)	Yes	No

Table 23. Uppercase translation truth table — ASIS option is specified

Profile	TYPETERM UCTRAN(YES)	TYPETERM UCTRAN(NO)
UCTRAN(YES)	No	No
UCTRAN(NO)	No	No

CRTE and uppercase translation

A description of how to initiate a CRTE session and the required input.

Initiate 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 24 on page 220.

Table 24. 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: If the transid CRTE is not entered in uppercase, 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 25 on page 220.

Table 25. 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: If the transid CRTE is not entered in uppercase, 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.

Be aware that when transaction routing from CICS Transaction Server for z/OS, Version 5 Release 4 to an earlier release of CICS that does not support transaction based uppercase translation, uppercase translation only occurs if it is specified in the 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 might occur.

This is because the command can override the 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 SNA logical units) 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.

CICS client virtual terminal

If the code page sent by a client is incorrect, this can lead to the entire screenful of data being incorrect. You must resolve this problem at the client end of operations.

The entire screenful of data might also be incorrect if the bit TCTSK_VIRTUAL_TERMINAL is not set on in the skeleton for the virtual terminal. The bit might have been overwritten, or not turned on when the virtual terminal was being created during CTIN processing.

Katakana terminals - mixed English and Katakana characters

If you are using a Katakana terminal, you might see some messages that contain 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 that are connected to your CICS system, specify **MSGCASE**=UPPER in the system initialization table to ensure that messages contain uppercase characters only.

The offline utilities DFHSTUP, DFHDU710, andDFHTU710 have an extra parameter to ensure that all output is translated to uppercase.

Data that is displayed incorrectly

Suggestions on how to deal with incorrect data values, partially missing data, early data that is overlaid by later data and data that is formatted incorrectly.

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 TYPETERM 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 PROFILE definition for the transaction running at the terminal, while default values for ALTSCREEN and ALTPAGE are allowed in the TYPETERM 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 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 TYPETERM 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 TYPETERM 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 PROFILE for the transaction running at the terminal.
- The PAGESIZE values included in the TYPETERM 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 and you will get 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.

Incorrect data is present on a VSAM data set

If 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 update 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 READ UPDATE.

An application does 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 command summary in IBM Knowledge Center. 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 main 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 produces 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.

1. Are there any messages explaining why there is no output?

   Look carefully in each of the transient data destinations CSMT, CSTL, and CDBC 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.
2. Can you use the terminal where the transaction should have started? See "Can you use the terminal where the transaction should have started?" on page 224 for the steps to follow.
3. 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 INQ TASK**.
   - If you have a suspended task, treat this as a "wait" problem. Use the techniques described in "Dealing with waits" on page 110 to investigate it further.
   - If you have a running task, it is likely to be looping. See "Dealing with loops" on page 196 to find out what to do next.

If you have obtained no output and the task is not in the system, read "No output - what to do if the task is not in the system" on page 224.

4. Did the task run? There are a variety of techniques for finding this information. These are described in "Techniques to find out whether a transaction started" on page 225

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 INQ TASK (or your site replacement) from the terminal.

If you cannot issue CEMT INQ 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 INQ 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 Chapter 3, "Distinguishing between waits, loops, and poor performance," on page 11 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 INQ TASK.

If you are able to issue CEMT INQ 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 INQ TASK`**. It is probably looping or waiting. See Chapter 3, "Distinguishing between waits, loops, and poor performance," on page 11 for advice about what to do next. If you do not see the task in the summary, go to "No output - what to do if the task is not in the system" on page 224.

No output - what to do if the task is not in the system

If you have obtained no output and CEMT INQ 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're not getting output on a printer, the reason could be 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.

Techniques to find out whether a transaction started

There are many different techniques for finding out whether a transaction started, or whether 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 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.

Before you begin

You need to use the CETR transaction to set up the right tracing options. See Using CICS trace for guidance about setting up trace options.

Procedure

1. Select special tracing for just your task, and disable tracing for all other tasks by setting the main system trace flag off.
2. 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. The components should always 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.
3. Now turn tracing on, and attempt to start your task.
4. When you are sure that the time has passed when the output should have appeared, stop tracing, and format the trace data set.

Results

If your transaction ran, you should see the following types of trace entries for your task and the programs associated with it:

- Loader domain, when it loaded your program, if the program was not already in main storage.
- Transaction manager, when it attached your task to the dispatcher.
- Dispatcher domain, when your task got its first dispatch. You might also see subsequent entries showing your task being suspended, and then resumed.
- 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 supplied transactions and EDF

You can use CICS-supplied transactions or the execution diagnostic facility (EDF) to find out whether a transaction started successfully.

Debug nonterminal transactions

You can use CEDX to debug nonterminal transactions. CICS intercepts the transaction specified on the **CEDX** *tranid* command, and displays the EDF diagnostic panels at the terminal at which the EDF command is issued. CEDX provides the same function and diagnostic display panels as CEDF, and the same basic rules for CEDF also apply to CEDX.

Debug transactions from a terminal

If the transaction that is being tested requires a terminal, you can use EDF. You must have 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 the command:

```
CEDF tttt
```

At the remaining terminal, enter the transaction, or sequence of transactions, that initiate the transaction that is being tested. Wait long enough for the transaction 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.

Debug transactions that use TD queues or TS queues

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 must be sure that no other transaction that might be executed while you are doing your testing performs the same action. The absence of such a change does not mean that the transaction did not run; it might have run incorrectly, so that the expected change was not made.

Debug transactions that write to files

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.

Using statistics

If nobody else is using the transaction that you are investigating, you can tell from CICS statistics whether the program has run.

About this task

You can use the **CEMT PERFORM STATISTICS RECORD** or **CEMT INQUIRE PROGRAM** command to obtain statistics that indicate whether the program has run. Use one of the following procedures.

Procedure

- Use the **CEMT PERFORM STATISTICS RECORD** command.
  a) Enter the command CEMT PERFORM STATISTICS RECORD, using the TRANSACTION option, before you test your transaction.

     ```
     CEMT PERFORM STATISTICS RECORD
     [TRANSACTION]
     ```

     Statistics on transactions that have been run are recorded in the SMF data set.
  b) Initiate the transaction and wait until it should have run.
  c) Repeat the **CEMT PERFORM STATISTICS RECORD** command to get a new set of statistics written to the SMF data set.
  d) Format the SMF data set for the APPLID that interests you, and look at the statistics recorded before and after you attempted to run the transaction.

     You can use the statistics utility program, DFHSTUP, to prepare and print reports offline using the data recorded in the SMF data set. The following control parameters can be useful:

```
SELECT APPLID=
COLLECTION TYPE=REQ
TIME START= ,STOP=
DATE START= ,STOP=
```

If the count for your transaction increased by 1, the transaction ran. If the count is unchanged, the transaction did not run.

- Use the **CEMT INQUIRE PROGRAM** command.
  a) Enter the command CEMT INQUIRE PROGRAM(*program*), where *program* is the program name.
  The resulting screen includes a USECOUNT value. This value is the number of times that the program has run since the start of the current CICS session.
  b) Initiate the transaction and wait until it should have run.
  c) Enter the **CEMT INQUIRE PROGRAM** command again. If the program has run, the USECOUNT value is incremented.

  The USECOUNT value is not reset as part of statistics. When the USECOUNT value reaches its maximum (2147483647), it does not increment but remains fixed at that maximum value.

Disabling the transaction

If your transaction requires a terminal, you can disable the transaction to test whether the problem is becauese of the terminal.

Use the following steps:

1. Use CEMT to disable the transaction that is being tested.
2. Initiate the transaction.

CICS issues the following message at the terminal where the transaction should 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.

AIDs are addressed from system entries with their forward and backward chain pointers at offset '0C' and '10' respectively. AIDs contain the following fields that can be useful in debugging.

AIDTYPE (X'2D')
Type of aid:

Content	Offset	Meaning
AIDBMS	X'80'	BMS AID
AIDPUT	X'50'	Start with data
AIDINT	X'40'	Start with no data
AIDTDP	X'10'	Transient data AID
AIDISC	X'08'	Queued allocate type AID
AIDCRRD	X'04'	Remote delete type AID

AIDSTATI (X'2E')
AID status indicator:

Content	Offset	Meaning
AIDPRIV	X'80'	Privileged allocate
AIDSENT	X'40'	This has been sent to the TOR by CRSR
AIDCANCL	X'20'	Cancel this AID
AIDROUTP	X'10'	Not yet routed to the AOR
AIDSHIPD	X'08'	Prevent duplicate send
AIDREMX	X'04'	AID for a remote transaction
AIDREMT	X'02'	AID for a remote terminal
AIDSTTSK	X'01'	Task already initiated

AID_TOR_NETNAME (X'65')
Netname of the owning region for a specific terminal

AID_TERMINAL_NETNAME (X'5D')
Netname of terminal

AIDDATID (X'34')
TS queue name holding the data.

AID_REROUTED (X'4E')
AID rerouted to a different TOR

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 TYPETERM definition.

Your transaction produces some output, but it is wrong

If your transaction produces no output at all, read "Your transaction produces no output at all" on page 223. 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.

For example, consider a transaction that reads records from a file, processes the information in the records, and displays the results on a terminal. 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.

Are 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.

Is 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 whether data is being corrupted by bad programming logic in the application, consider the flow of data through the transaction.

You can determine the flow of data through your transaction by "desk checking", or by using the interactive tools and tracing techniques supplied by CICS.

Desk checking your source code is sometimes best done with the help of another programmer who is not familiar with the program. Such a person might see weaknesses in the code that you have overlooked.

You can use interactive tools to see how the data values being manipulated by your program change as the transaction proceeds.

- CEDF is a 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 ASKTIME at critical points in your program, to see if certain paths are taken, and to check program storage values.
- You can use CECI 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.
- You can use CEBR 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.

  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.

You can use user tracing 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. For example, you can 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, and 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 Application development reference.

You can use CSFE storage freeze to freeze the storage associated with a terminal or a transaction so that it is not released by a freemain request 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 released by a freemain request 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.

Dealing with storage violations

A storage violation occurs when a transaction attempts to modify storage that it does not own.

The following topics describe how to deal with storage violations:

Avoiding storage violations

CICS provides three facilities that help to prevent storage violations.

About this task

CICS subsystem storage protection
prevents user application programs from directly overwriting CICS code and control blocks.

Transaction isolation
prevents a user transaction from directly overwriting user application storage of other transactions.

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 other transactions' storage.
- An application program could be defined with EXECKEY(CICS) and could thus modify CICS storage and other transactions' storage.
- 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 Writing a program error program.

If a storage violation occurs in your system, please read the rest of this section.

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 Xcode')
 has been detected by module modname
```

If you have received this message, turn first to the description of message DFHSM0102 in CICS messages to see an explanation of the message, and then to Trace entries to see an explanation of the exception trace point ID, X'*code*'. This tells you how CICS detected the storage violation. Then return to this section, and read "CICS has detected a storage violation" on page 232.

Storage violations not detected by CICS are less easy to identify. They can cause almost any sort of symptom. Typically, you might 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 236.

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 FREEMAIN command to release an individual element of TIOA storage, at least as far as the target element. It also checks the chains when it releases the storage, by using a FREEMAIN request, 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 release that element of storage by using a FREEMAIN request. It also checks the chains when it releases all the storage , by using a FREEMAIN request, 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 33 on page 233, 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 33 on page 233.

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 236.

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.

User task issues GETMAIN. Storage element is obtained. The leading and trailing check zones match.

Task writes data, overlaying the trailing storage check zone
When the task ends, CICS attempts to FREEMAIN the storage element, but finds that the two storage check elements are not identical. CICS issues an error message and continues. The corrupted element remains unchanged, and cannot be reused, unless storage recovery is on.

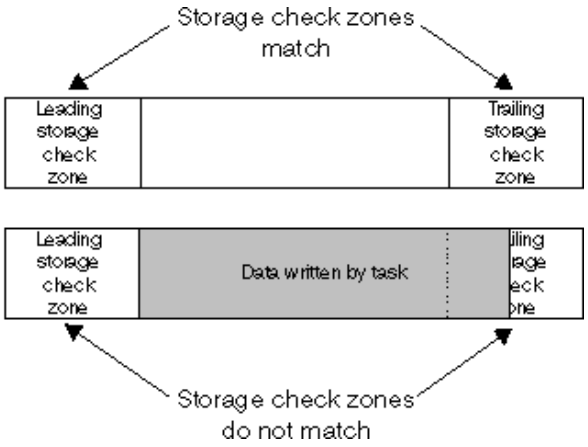


Figure 33. How user-storage violations are committed and detected

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 in the SIT), the corrupted SAAs or check zones are repaired and the transaction continues. See “Storage recovery” on page 238.

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. This is in addition to the SM0102 system dump.

If you received a transaction abend message, read “What the transaction abend message can tell you” on page 233. Otherwise, go on to “What the CICS system dump can tell you” on page 234.

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 probably is 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 Using CICS trace.)

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 parameter.

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 26 on page 235 shows the CSFE DEBUG options and their effects. Table 27 on page 235 shows the startup overrides that have the same effects.

Table 26. Effects of the CSFE DEBUG transaction	
CSFE syntax	**Effect**
`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 27. Effects of the CHKSTSK and CHKSTRM overrides	
Override	**Effect**
`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 Using CICS trace.

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 Table 7 on page 72.)

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. A description of the exception trace point ID, and the data it contains, is in Trace entries.

Format the system dump using the formatting keyword TR, to get the internal trace table. Locate the exception trace entry made when the storage violation was detected, near the end of the table. Now 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.

If you have used the CHKSTSK=CURRENT option, you can locate the occurrence of the storage violation only with reference to the last old-style trace entry for the current task.

You need to 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 237.

Storage violations that affect innocent transactions

CICS does not usually detect storage violations that affect innocent transactions; that is, transactions that do not cause the violation. However, CICS sometimes 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 that writes data to a part of the DSAs that it does not own, or possibly that releases such an area by using a freemain request. The transaction might obtain the area by using a getmain request and then release it by using a freemain request before writing the data, or an application might not maintain addressability correctly in another way. Another possible reason is that a transaction posted an event control block (ECB) after the task that was waiting on it was canceled.

Storage violations that affect innocent transactions are generally more difficult to resolve than those that CICS detects. You might not become aware of them until some time after they occur, so you need a long history of system activity to find out their cause.

A strategy for storage violations affecting innocent transactions

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 could help you to identify the transaction, program, or routine that caused the error. If it does not provide the clue you need, your strategy should be to 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 mean that a very large trace table is needed. Internal trace is not suitable, because it wraps when it is full and it then overwrites important trace entries.

Auxiliary trace is a suitable destination for recording long periods of system activity, because it is possible to specify very large auxiliary trace data sets, and they do not wrap when they are 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.

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 special tracing for all user transactions and all terminals, and disable standard tracing by setting the main system trace flag off.
2. Use the CETR transaction to set up the tracing options, and select auxiliary trace as the trace destination. 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.
3. 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.
4. 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.
5. 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.
6. Having found the identity of the transaction, 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 might suggest coding a global trap/trace exit to detect the storage violation.

Programming errors that can cause storage violations

A number of commonly occurring programming errors can cause storage violations.

1. Failing to obtain sufficient storage when using a getmain request. 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 released by using a freemain request.

   When a task releases an area that it has been addressing by using a freemain request, 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 might corrupt data that belongs to an unrelated activity if the area that was 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.

Dealing with external CICS interface (EXCI) problems

The following CICS messages support the external CICS interface:

```
DFHIR3799

DFHEX0001         DFHEX0011         DFHEX0400
DFHEX0002         DFHEX0012
DFHEX0003         DFHEX0013
DFHEX0004         DFHEX0014
DFHEX0005         DFHEX0015
DFHEX0010         DFHEX0016
```

Messages DFH5502W and DFH5503E include support for the external CICS interface facility.

This facility is also supported by two translator messages, DFH7004I and DFH7005I. For full details of all CICS messages, see CICS messages.

The external CICS interface outputs trace to two destinations: an internal trace table and an external MVS GTF data set. The internal trace table resides in the non-CICS MVS batch region. Trace data is formatted and included in any dumps produced by the external CICS interface.

Trace entries are issued by the external trace interface destined for the internal trace table or an MVS GTF data set. They are listed in Trace entries

The external CICS interface produces MVS SYSM dumps for some error conditions and MVS SDUMPs for other, more serious conditions. These dumps contain all the external CICS interface control blocks, as well as trace entries. You can use IPCS to format these dumps.

A user-replaceable module, 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.

Dealing with TCP/IP connectivity problems

If you are experiencing TCP/IP connectivity problems, read the following information to find out how to diagnose the problem.

Before you begin

Ensure that TCP/IP services are active by specifying TCPIP=YES as a system initialization parameter. Ensure that your TCP/IP resource definitions are defined, installed and open. For more information about configuring TCP/IP, see Using TCP/IP in a CICS region.

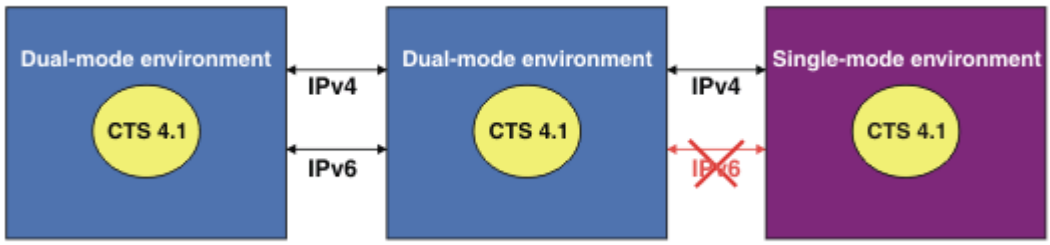
About this task

You might be having problems either because you have changed the HOST attribute in your resources to allow IPv6 traffic, or you might have a problem and none of your configuration settings have changed, for example, because there have been external changes that have affected your settings. Investigate the source region and target region to find out why you have problems connecting.

Procedure

1. For each region, determine if you are operating in a single-mode (IPv4) environment or a dual-mode (IPv4 and IPv6) environment.

   Common problems related to IPv6 are because an environment does not support IPv6 and an IPv6 address has been used to attempt a connection to that environment. The figure shows that two dual-mode CICS TS 4.1 environments can communicate using either IPv4 or IPv6 addressing. A single-mode CICS TS 4.1 environment is also connected, but can communicate using IPv4 only.



   For more information about single and dual-mode environments, see Understanding IPv6 and CICS. The following problems are examples of messages received when there is a mismatch of IPv4 and IPv6 environments:

   a) If the TCPIPSERVICE definition in the listening CICS region specifies an IPv6 address in the HOST attribute (or a *hostname* that resolves to an IPv6 address) and either CICS region is operating in a single mode environment, the listening CICS region might receive a DFHAM4907W message "Opening TCPIPSERVICE XXXX has failed because the IP address is not known". One of the situations when this message is issued is because a region operating in a single-mode environment does not have IPv6 capability.

   b) If a CICS region operating in a dual-mode environment issues a **WEB OPEN** command, specifying an IPv6 address (or a *hostname* that resolves to an IPv6 address) to a listening CICS region that is operating in a single-mode environment, the CICS region issuing the command might receive a NOTFND error code with a RESP2 value of 20 (host name is not resolved by name server or the format of the host option is incorrect). This message can be issued because the dual-mode environment has attempted to connect using IPv6 and the single-mode region does not have IPv6 capability.

   c) Check that you have not explicitly specified IPv6 address information in the HOST option of your resources (or a *hostname* that resolves to an IPv6 address), unless you know that you operate in a CICS TS 4.1 dual-mode environment, and that any remote regions you want to connect to CICS TS 4.1 also operate dual-mode environments.

      You might receive message DFHIS1007 if an invalid attempt was made to use an IPv6 address.

d) Check that an attempt to install and open the TCPIPSERVICE resource was successful.

   If message DFHSO0110 is issued, it might be that an IPv6 address or a *hostname* that resolves to an IPv6 address has been specified, and there is no dual-mode capability to support the request.

e) If the TCPIPSERVICE resource opens successfully, check message DFHSO0107, which returns the IP address that was used to connect.

   This will give you more information on the type of address used.

f) Check that the address you entered is being displayed in the format that you expect.

   If you have specified a low IPv6 address, that is, the address has leading zeros in the first six or more segments, CICS stores and displays the IPv6 address in IPv4 format. Here are some examples of how CICS displays low IPv6 addresses:

Table 28. How CICS handles valid low IPv6 addresses

Specified valid IPv6 address	**How CICS stores and displays the address**
`::9`	`0.0.0.9`
`::10`	`0.0.0.16`
`::10A`	`0.0.0.10`
`::ABCD`	`0.0.171.205`
`::FFFF`	`0.0.255.255`
`::FFFF:9`	`255.255.0.9`
`::FFFF:10`	`255.255.0.16`
`::FFFF:10A`	`255.255.0.10`
`::FFFF:ABCD`	`255.255.171.205`
`::FFFF:FFFF`	`255.255.255.255`

2. For both regions, inquire on the resource which relates to the protocol that you are using (for example, URIMAP) and compare to the listening TCPIPSERVICE resource. If you are using the IPIC protocol, inquire on the both sets of complementary IPCONN and TCPIPSERVICE resources to compare attributes. For example, inquire on the following TCP/IP resource attributes:

   a) Inquire on the HOST option to determine the host name, IPv4, or IPv6 address specified for the region.

      If you have an IPv6 address or a *hostname* that resolves to an IPv6 address in the HOST option, ensure that both CICS regions are operating in dual-mode environments.

   b) If there is a host name in the HOST option, this might resolve to an IPv6 address at the domain name server. Trace points SO 0436 and SO 0437 give details of resolved addresses, or consult your system support representative to find out the IP addresses that have been configured at the domain name server for this port.

3. Look for any error messages or TCP/IP connection information that might indicate the cause of the connectivity problem; for example, a port sharing conflict or a problem because the task has not run properly.

   You can inquire on socket application data (ApplData) for each TCP socket owned by CICS to correlate TCP/IP connections with the CICS regions and transactions using them. ApplData is also available using Netstat reports. For more information about CICS ApplData, see TCP/IP management and control.

4. Modify any resources that have not been correctly defined.
5. Verify that traffic is flowing across the connection.

What to do next

The network connection is available. If the connection is not available, contact your Network Administrator and request an investigation into the network components that are being used to connect your CICS regions.

Dealing with log manager problems

The CICS log manager uses services provided by the MVS logger to support the logging and journaling of data to the CICS system log, forward recovery logs, autojournals, and user journals.

This section contains the following topics:

Categories of problem

The following categories of problem (in order of ascending impact on the user) may be encountered by the CICS log manager.

1. Those problems within the MVS logger that the MVS logger resolves for itself. CICS has no involvement in this category and might only experience the problem as an increase in response times.
2. Where the MVS logger is unable to satisfy the CICS log manager's request immediately. This problem state can be encountered:
   - For a log stream that uses a coupling facility structure, on a "STRUCTURE FULL" condition, where the coupling facility has reached its capacity before offloading data to DASD. This state may also be encountered during the rebuilding of a coupling facility structure.
   - For a DASD-only log stream, on a 'STAGING DATA SET FULL' condition, where the staging data set has reached its capacity before offloading data to secondary storage.

   If either of these conditions occur, CICS issues message DFHLG0771 (for a general log) or DFHLG0777 (for a system log). The CICS log manager retries the request every three seconds until the request is satisfied. Typically, this can take up to a minute.
3. If the MVS logger fails, CICS is abended. If the system log has not been damaged, a subsequent emergency restart of CICS should succeed.
4. If a return code implies that the CICS system log has been damaged, CICS is quiesced, meaning transactions are allowed to run to completion as far as possible, with no further records being written to the system log. To get CICS back into production, you must perform an initial start. However, before doing so you may want to perform a *diagnostic run*, to gather information for problem diagnosis - see "Dealing with a corrupt system log" on page 253.

   If a return code implies damage to a forward recovery log or autojournal, all files using the log stream are quiesced and their transactions run to completion. Message DFHFC4800, DFHFC4801, or DFHFC4802 is issued. User transactions writing journal records to the log stream experience a write error. For a forward recovery log, before you can continue to use the log stream, you must:

   a. Take an image copy of **all** data sets referencing the log stream.
   b. Redefine the log stream.
   c. Unquiesce the data sets using the affected logs. You may then explicitly open the files but they open automatically at the first READ or WRITE if they are in a CLOSED ENABLED state after the unquiesce.

   For an autojournal, before you can continue to use the log stream, you must:

   a. Try to read and recover data from the damaged autojournal.

b. Redefine the log stream.

Exceeding the capacity of a log stream

The MVS logger imposes a limit on the number of data sets per log stream. In practice, this is unlikely to be a problem.

System log

You are strongly recommended to allow the CICS log manager to manage the size of the system log. If you do so, you do not need to worry about the data set limit being exceeded.

In the unlikely event that you need to retain data beyond the time it would be deleted by CICS, see Managing auxiliary storage for advice on how to define the system log.

General logs

If a journal write to a user journal fails because the data set limit is reached, you must delete the tail of the log, or archive it, before you can use the **SET JOURNALNAME** command to open the journal and make it available for use again. For an example of how to do this, see Examples of using DFHJUP.

- The number of data sets per log stream recognized by the MVS logger is several million. In normal circumstances, you do not need to be concerned about the limit being exceeded.
- You can cause redundant data to be deleted from log streams automatically, after a specified period. To arrange this for general log streams, define the logs to MVS with AUTODELETE(YES) and RETPD(dddd), where dddd is the number of days for which data is to be retained. This causes the MVS logger to delete an entire log data set when all the data in it is older than the retention period (RETPD) specified for the log stream.

How CICS checks for the availability of the MVS logger

At intervals, CICS itself checks for the availability of the MVS logger. It uses one of two procedures to perform these checks, depending on your operating system.

CICS checks by querying the system log connection status. If the check fails, CICS either abends or quiesces, depending on the returned MVS logger reason code.

The interval at which CICS checks for the availability of the MVS logger varies, depending on the amount of system logging activity in the CICS region. The first check is made after CICS has not made contact with the MVS logger for 10 seconds. If CICS continues to perform no system logging after the first check, the interval between checks doubles each time, up to a maximum of 600 seconds. If CICS makes contact with the MVS logger at any point, the interval between checks is halved, down to a minimum of 10 seconds.

The checking interval can be affected by the exit time interval specified in the **ICV** system initialization parameter:

- If the value specified in the **ICV** system initialization parameter is less than 10 seconds, it has no effect on the checking interval.
- If the value specified in the **ICV** system initialization parameter is greater than 10 seconds but less than 600 seconds, the checking interval varies between the value specified in the **ICV** system initialization parameter, and 600 seconds. The first check is made after an interval corresponding to the value in the **ICV** system initialization parameter, instead of being made after 10 seconds. The minimum checking interval is the value in the **ICV** system initialization parameter.
- If the value specified in the **ICV** system initialization parameter is greater than 600 seconds, the checking interval does not vary, and always corresponds to the value in the **ICV** system initialization parameter.

For more information about this parameter, see ICV system initialization parameter.

Use the statistics field IGXQUERY in the CICS log manager statistics to monitor the number of checks that CICS makes for the availability of the MVS logger.

Some conditions that cause CICS log manager error messages

Problems in the CICS log manager and its interface with the MVS system logger can arise from various conditions, as a result of which the CICS region can fail because of loss of access to its system log. You need to be aware of the more common conditions, which (although very rare) result in a failure in this component of CICS.

In order to understand what has happened in a particular failure, it is helpful to look at the various combinations of messages that can be issued by CICS in different error situations. This approach is useful in ensuring that you gather the necessary diagnostic information at the time of the failure, to enable accurate problem determination and resolution. It also helps to ensure a rapid restart of the CICS region, with a full appreciation of the possible impact on data integrity.

The following CICS log manager messages cover some of the CICS logger failure situations. The more common message combinations are as follows:

DFHLG0772, DFHLG0800, and DFHLG0738
DFHLG0772, DFHLG0800, DFHLG0736, and DFHLG0741
DFHLG0772 and DFHLG0740
DFHLG0772 and DFHLG0734
DFHLG0002 and DFHLG0734

Note that if messages DFHLG0736, DFHLG0738, or DFHLG0740 are issued, CICS recovery manager sets its global catalog type-of-start override record to AUTODIAG. For more information, see "Restarting CICS after a system log failure" on page 246.

Note: For details of all the return and reason codes for the MVS logger macros, see z/OS MVS Programming: Authorized Assembler Services Reference (Volume 2).

Message DFHLG0772

DFHLG0772 is the first message CICS issues if the MVS logger returns an exception condition in response to a call to an IXGCONN, IXGWRITE, IXGBRWSE, or IXGDELET operation.

The MVS logger return and reason codes for the failure are given in the message, together with the name of the call and the attributes of the log stream being accessed at the time. Message DFHLG0772 is followed by one or more other messages when CICS has determined the extent of the problem.

CICS takes a system dump at the time the message is issued. This is the primary source of information for determining the cause of the error, and a dump from a DFHLG0772 should *not* be suppressed. See "Setting a SLIP trap" on page 251 for information on how to capture dumps of the MVS system logger address space and the coupling facility structure at the same time. These three pieces of documentation are essential if you refer the problem to IBM service. You are also recommended to run the DFHJUP utility to obtain printed output from the DFHLOG and DFHSHUNT system log streams before you restart a failed region.

If CICS decides the data integrity is compromised, or the problem is too serious to allow continued operation, it marks the system log as broken. CICS then begins automatic action to shut itself down.

Log block not found with DFHLG0800, DFHLG0736 and DFHLG0741

If CICS issues DFHLG0772 with return code 8 and reason code `IxgRsnCodeNoBlock` (`00000804`), it means that the MVS logger could not find a log block requested by CICS, because the log data is missing from the log stream.

If this occurs *after* initialization is complete, CICS issues messages DFHLG0800, DFHLG0736 and at least one DFHLG0741:

DFHLG0800

This provides further diagnostic information to complement the system dump captured with the preceding DFHLG0772 message.

DFHLG0800 gives the log stream block ID of the requested block and the block ID of the chain history point of the log block chain being read by CICS. The DFHLG0800 message is one of the

most important pieces of diagnostic information when investigating a failure caused by an 00000804 reason code.

DFHLG0736
This message informs you that CICS is performing a normal shutdown, issued by system task CSQC, following the DFHLG0772 and DFHLG0800 messages.

The task that was executing when the error occurred is abended but cannot perform backout because of the failure (see message DFHLG0741).

Performing a normal shutdown allows all other existing tasks unaffected by the error to complete normally, but prevents new work from starting. Note that this phase of processing is exceptional in that CICS cannot make any use of the system log either for reading or writing, and therefore updates performed by the tasks running in this phase are not recoverable: if one of these tasks abends, any updates it makes cannot be backed out by CICS. If a task does abend in this phase, capture details of the task from the associated message DFHLG0741 (see below).

If the other existing tasks complete normally with a successful syncpoint, CICS does not need to read the log for any data that may have been written for these tasks before the system log failed. These successfully completed tasks are unaffected by the failure, even though the log is marked as "broken".

DFHLG0741
This message follows DFHLG0736, and identifies the task ID, the transaction ID and the terminal ID associated with the task. There must be at least one task that has attempted to read the system log—the task that issued the log manager request that led to the DFHLG0772, DFHLG0800, and DFHLG0736 set of messages being issued. CICS suspends the task indefinitely with resource type LGFREVER (meaning "logger wait forever"). The current UOW in the suspended task cannot be allowed to complete and commit its changes. This decision is taken because log data relating to the UOW has been lost. Similarly, the UOW cannot be backed out by dynamic transaction backout, because the required before images cannot be read from the system log.

CICS issues a DFHLG0741 message for each task affected in this way. If any other in-flight tasks attempt backout (by issuing a SYNCPOINT ROLLBACK or ABEND command, or by failing and being abended by CICS), these also are suspended LGFREVER. They are in the same position as the task that triggered the DFHLG0736 message. That is, they are attempting to retrieve log data from the system log, and CICS cannot guarantee the integrity of the system log because some of the log data is not accessible by the MVS logger.

Note: The quiesce of CICS initiated with message DFHLG0736 continues until the in-flight tasks on the system complete, either successfully by committing their updates, or by abending. Those tasks that attempt a backout are suspended forever. CICS, therefore, is unable to complete a normal shutdown operation and hangs, requiring intervention to be terminated. This intervention can be by one of the following:

- Operator action
- The shutdown assist transaction
- A CICS monitor package.

The intervention is required because there is at least one task suspended indefinitely in an LGFREVER wait.

After DFHLG0800, DFHLG0736, and DFHLG0741, ensure that you perform a diagnostic start, followed by an initial start when you have successfully captured the diagnostics. See “Restarting CICS after a system log failure” on page 246 for details.

Log block not found with DFHLG0800 and DFHLG0738
If CICS issues DFHLG0772 with return code 8 and reason code IxgRsnCodeNoBlock (00000804), it means that the MVS logger could not find a log block requested by CICS, because the log data is missing from the log stream.

If this occurs *during* initialization, CICS issues messages DFHLG0800 and DFHLG0738:

DFHLG0800
This provides further diagnostic information to complement the system dump captured with the preceding DFHLG0772 message.

DFHLG0800 gives the log stream block ID of the requested block and the block ID of the chain history point of the log block chain being read by CICS. The DFHLG0800 message is one of the most important pieces of diagnostic information when investigating a failure caused by an `00000804` reason code.

DFHLG0738
This informs you that CICS cannot continue initializing and is terminating. The MVS logger has failed to retrieve log data in which CICS has an interest during the restart, and because of this CICS cannot rely on the integrity of the system log.

After DFHLG0800 and DFHLG0738, ensure that you perform a diagnostic start, followed by an initial start when you have successfully captured the diagnostics. See "Restarting CICS after a system log failure" on page 246 for details.

Loss of log data with DFHLG0740
If DFHLG0772 gives the MVS system logger reason code as `IxgRsnCodeLossOfDataGap` (`0000084B`), it is followed by DFHLG0740.

DFHLG0740
This explains that a write request to the system log completed successfully, but the MVS logger has detected that previously hardened log data has since been lost from the log. Therefore, the integrity of the system log is suspect, because CICS might need to refer to the missing log data at some point in the future either for dynamic transaction backout of a UOW, or system recovery on either a cold, warm or emergency restart.

CICS initiates a quiesce of the system in the same way as for a DFHLG0736 message. If all in-flight tasks complete normally and commit their changes, they terminate successfully with no need to refer to the system log, ensuring data integrity of local resources is maintained. However, as in the case of DFHLG0741 (see above) if any in-flight tasks attempt backout (by issuing a SYNCPOINT ROLLBACK or ABEND command, or by failing and being abended by CICS), these are suspended with resource type LGFREVER. Any transactions that fail to complete before shutdown must be recovered by some other way before starting CICS again.

CICS forces the next restart to be an initial start, because this is the only type of restart that has no interest in any log data previously stored on the system log

Log error with DFHLG0734
If CICS issues DFHLG0772 with a reason code other than block not found (`IxgRsnCodeNoBlock`) or loss of log data (`IxgRsnCodeLossOfDataGap`), it issues message DFHLG0734.

DFHLG0734
This indicates a severe exception condition, indicated by the reason code in the preceding DFHLG0772 message and CICS immediately terminates. The problem should be investigated and the error corrected before restarting CICS.

Under this abnormal termination CICS does attempt to allow in-flighttasks to complete. In this case, CICS does not force the next type of start to be an initial start. The type-of-restart indicator in the recovery manager control record is set to "emergency restart needed," to ensure CICS performs an emergency restart. This is because the nature of this error and its resolution should allow a CICS emergency restart to restore the system to a committed state. This assumes the system log remains intact and is accessible to CICS when you perform the restart.

Message DFHLG0002

This is a general message that is issued when a severe error has occurred within the CICS log manager domain. The module in error is identified in the message, together with the unique error code value. CICS takes a system dump to allow problem determination of the severe error condition.

This is usually followed by another message, typically DFHLG0734.

Severe log manager error with DFHLG0734

If CICS issues DFHLG0002, but determines that an emergency restart may resolve the error and successfully recover in-flight tasks, CICS issues DFHLG0734.

DFHLG0734
This indicates a severe exception condition, indicated by the reason code in the preceding DFHLG0002 message and CICS immediately terminates. The problem should be investigated and the error corrected before restarting CICS.

The type-of-restart indicator in the recovery manager control record is set to "emergency restart needed," to ensure CICS performs an emergency restart. This is because the nature of this error and its resolution could allow a CICS emergency restart to restore the system to a committed state. This assumes the system log remains intact and is accessible to CICS when you perform the restart.

Restarting CICS after a system log failure

When you restart CICS with START=AUTO after a failure following DFHLG0736, DFHLG0738, or DFHLG0740, CICS initializes for a diagnostic run only.

About this task

On a diagnostic run, CICS produces a dump of the CICS region state, retrieved from the CICS system log and then terminates. On a diagnostic run, CICS performs no recovery work and no new work. This situation persists until you start the region with an initial start.

A diagnostic run produces diagnostics for investigation by IBM Service. For example, if DFHLG0772 reported return code 8 and reason code `IxgRsnCodeNoBlock` (`00000804`), but the associated system dump is lost, the diagnostic run reproduces the dump (assuming the `00000804` condition is a solid failure).

For information about the AUTODIAG type-of-start override record, see Specifying parameters for DFHRMUTL. For more details of a diagnostic run, see “Dealing with a corrupt system log” on page 253.

When you have obtained the required diagnostics and are ready to restart the region with the broken system log, you can do so only with an initial start. You can do this either by running the DFHRMUTL utility with the SET_AUTO_START=AUTOINIT parameter, or by specifying START=INITIAL as a system initialization parameter.

Procedure

- Run the DFHRMUTL utility with the **SET_AUTO_START**=AUTOINIT parameter.
- Alternatively, specify the **START**=INITIAL as a system initialization parameter.

Results

An initial start is the only form of CICS startup that does refer to log data written during the previous run. It is the only restart that is possible in these circumstances.

Diagnosing problems in the MVS logger

Extended waits by the CICS log manager can be caused by problems within the MVS logger or other areas of MVS. You can investigate these by looking at the MVS console messages.

About this task

Look at the following:

- "Console messages and dumps" on page 247
- "GRS resource contention" on page 247
- "Checking coupling facility structure and couple data set status" on page 248
- "Checking log stream status" on page 249
- "SMF and RMF statistics" on page 251
- "Obtaining MVS logger and coupling facility dumps" on page 251
- "Restarting the MVS logger address space" on page 253

Console messages and dumps

Look for:

- Outstanding WTOR messages
- IXGxxx messages
- Allocation, catalog and HSM error messages
- IO errors for log stream data sets and LOGR couple data sets
- IXCxxx messages, indicating problems with the coupling facility structure or couple data sets.
- 1C5 abends, and other abends from the IXGLOGR address space.

Log stream data sets are of the form `IXGLOGR.stream_name.Annnnnnn`. The high level qualifier (IXGLOGR) may be different if the HLQ parameter was specified when the log stream was defined.

Explanations of MVS logger reason codes which are shown in CICS and MVS messages and traces are in the IXGCON macro and in z/OS MVS Programming: Assembler Services Reference ABE-HSP .

GRS resource contention

To check GRS resource contention by displaying GRS enqueues and latch usage on all machines in the sysplex, issue either of the following MVS commands.

The RO *ALL phrase means that the command goes to all systems in the sysplex:

```
RO *ALL,D GRS,C

RO *ALL,D GRS,RES=(SYSZLOGR,*)
```

A normal response looks like:

```
D GRS,C

ISG020I 12.06.49 GRS STATUS 647
NO ENQ CONTENTION EXISTS
NO LATCH CONTENTION EXISTS

D GRS,RES=(SYSZLOGR,*)

ISG020I 14.04.28 GRS STATUS 952
NO REQUESTORS FOR RESOURCE  SYSZLOGR *
```

A response showing GRS contention looks like this. You may also see latch set name SYS.IXGLOGER_MISC:

```
D GRS,C

ISG020I 12.06.31 GRS STATUS 619
LATCH SET NAME:  SYS.IXGLOGER_LCBVT
CREATOR JOBNAME: IXGLOGR   CREATOR ASID: 0202
  LATCH NUMBER:  7
    REQUESTOR  ASID  EXC/SHR    OWN/WAIT
    IXGLOGR    0202  EXCLUSIVE  OWN
    IXGLOGR    0202  SHARED     WAIT

D GRS,RES=(SYSZLOGR,*)

ISG020I 19.58.33 GRS STATUS 374
S=STEP    SYSZLOGR  91
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV26      MSLDELC1           002F       008F6370 EXCLUSIVE   OWN
S=STEP    SYSZLOGR  93
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV26      MSLWRTC1           002E       008DED90 EXCLUSIVE   OWN
MV26      MSLWRTC1           002E       008DB990 EXCLUSIVE   WAIT
MV26      MSLWRTC1           002E       008DB700 EXCLUSIVE   WAIT
MV26      MSLWRTC1           002E       008F60C8 EXCLUSIVE   WAIT
S=SYSTEMS SYSZLOGR LPAYROL.TESTLOG.TLOG1
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV27      IXGLOGR            0011       008F7398 EXCLUSIVE   OWN
MV26      IXGLOGR            0011       008F7398 EXCLUSIVE   WAIT
```

This shows which tasks (that is, MVS TCBs) have exclusive enqueues on the log streams, and which tasks are waiting for them. It is quite normal for enqueues and latches to be obtained, occasionally with contention. They are indications of a problem only if they last for more than a minute or so.

Long term enqueuing on the SYSZLOGR resource can be a sign of problems even if there is no contention.

You can choose to display only those log streams exclusively enqueued on by CICS jobs in the sysplex. Issue the following MVS command:

```
D GRS,RES=(DFHSTRM,*)
```

A typical response to this command looks like this:

```
ISG020I 14.51.28 GRS STATUS 541
S=SYSTEMS DFHSTRM  PAYROL.CICSVR.DFHLGLOG
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL91           0042       007D9108   SHARE     OWN
MV29      PAYROL93           0044       007D9138   SHARE     OWN
S=SYSTEMS DFHSTRM  PAYROL.FWDRECOV.UTL3
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL91           0042       007D9108   SHARE     OWN
MV29      PAYROL93           0044       007D9138   SHARE     OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET1.DFHJ02
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL91           0042       007D9108   SHARE     OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET1.DFHLOG
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL91           0042       007D9108 EXCLUSIVE   OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET1.DFHSHUNT
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL91           0042       007D9108 EXCLUSIVE   OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET3.DFHJ02
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL93           0044       007D9138   SHARE     OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET3.DFHLOG
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL93           0044       007D9138 EXCLUSIVE   OWN
S=SYSTEMS DFHSTRM  PAYROL.IYK8ZET3.DFHSHUNT
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR  OWN/WAIT
MV29      PAYROL93           0044       007D9138 EXCLUSIVE   OWN
```

Checking coupling facility structure and couple data set status

To display the MVS logger couple data set status, issue the following MVS command:

```
D XCF,CPL,TYPE=LOGR
```

A normal response looks like this:

```
D XCF,CPL,TYPE=LOGR
IXC358I  14.47.51  DISPLAY XCF 391
LOGR COUPLE DATA SETS
PRIMARY    DSN: SYS1.SYSPLEX2.SEQ26.PLOGR
           VOLSER: P2SS05     DEVN: 230D
           FORMAT TOD         MAXSYSTEM
           12/20/95 09:25:48          8
ALTERNATE  DSN: SYS1.SYSPLEX2.SEQ26.ALOGR
           VOLSER: P2SS06     DEVN: 2C10
           FORMAT TOD         MAXSYSTEM
           12/20/95 09:27:45          8
LOGR IN USE BY ALL SYSTEMS
```

If the response shows that LOGR is not in use by all systems, there may be a problem to investigate. Look for IXCxxx messages which might indicate the cause of the problem and issue the following command to attempt reconnection to the couple data set:

```
SETXCF CPL,TYPE=(LOGR),PCOUPLE=(couple_dataset_name)
```

To display all structures with Failed_persistent connections, issue the following MVS command:

```
D XCF,STR,STRNM=*,STATUS=FPCONN
```

The MVS logger should resolve any failed connections.

Checking log stream status

To display information about the status of CICS log streams, a batch job should issue the IXCMIAPU command:

```
LIST LOGSTREAM NAME(streamname) DETAIL(YES)
```

You can use wildcards to select multiple log streams. For example, the following job produces a report on the system log streams for CICS region IYLX4:

```
//IYLXLIST JOB NOTIFY=WILLIN,MSGCLASS=A
//LOGLIST  EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A,DCB=RECFM=FBA
//SYSIN    DD *
  DATA TYPE(LOGR) REPORT(NO)
  LIST LOGSTREAM NAME(WILLIN.IYLX4.DFH*) DETAIL(YES)
```

Figure 34 on page 250 shows a typical response to this command, with system logs streams for CICS region IYXL4.

```
  LOGSTREAM NAME(WILLIN.IYLX4.DFHLOG) STRUCTNAME() LS_DATACLAS()
         LS_MGMTCLAS() LS_STORCLAS() HLQ(IXGLOGR) MODEL(NO) LS_SIZE(0)
         STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
         LOWOFFLOAD(40) HIGHOFFLOAD(85) STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
         RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO)
         DASDONLY(YES)
         MAXBUFSIZE(64000)
       LOG STREAM ATTRIBUTES:
         User Data:
          00000000000000000000000000000000000000000000000000000000000000000000
          00000000000000000000000000000000000000000000000000000000000000000000

      LOG STREAM CONNECTION INFO:
         SYSTEMS CONNECTED: 1
         SYSTEM    STRUCTURE         CON CONNECTION  CONNECTION
        NAME      VERSION           ID  VERSION     STATE
        --------  ----------------  --  ----------  ----------
        MV28      0000000000000000  00  00000000    N/A

       LOG STREAM DATA SET INFO:
         DATA SET NAMES IN USE: IXGLOGR.WILLIN.IYLX4.DFHLOG.<SEQ#>
         Ext.   <SEQ#>    Lowest Blockid    Highest GMT        Highest Local
        -----  --------  ----------------  -----------------  ---------------
       *00001  A0000007  0000000000496BAB  07/18/97 08:29:13  07/18/97 09:29:

       NUMBER OF DATA SETS IN LOG STREAM: 1
      POSSIBLE ORPHANED LOG STREAM DATA SETS:
         DATA SET NAMES:
        -----------------------------------------
        IXGLOGR.WILLIN.IYLX4.DFHLOG.A0000037
        IXGLOGR.WILLIN.IYLX4.DFHLOG.A0000404

        NUMBER OF POSSIBLE ORPHANED LOG STREAM DATA SETS: 2
```

```
LOGSTREAM NAME(WILLIN.IYLX4.DFHSHUNT) STRUCTNAME() LS_DATACLAS()
          LS_MGMTCLAS() LS_STORCLAS() HLQ(IXGLOGR) MODEL(NO) LS_SIZE(0)
          STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
          LOWOFFLOAD(0) HIGHOFFLOAD(80) STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
          RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO)
          DASDONLY(YES)
          MAXBUFSIZE(64000)
       LOG STREAM ATTRIBUTES:
         User Data:
          00000000000000000000000000000000000000000000000000000000000000000000
          00000000000000000000000000000000000000000000000000000000000000000000

       LOG STREAM CONNECTION INFO:
         SYSTEMS CONNECTED: 1
         SYSTEM    STRUCTURE         CON CONNECTION  CONNECTION
        NAME      VERSION           ID  VERSION     STATE
        --------  ----------------  --  ----------  ----------
        MV28      0000000000000000  00  00000000    N/A

      LOG STREAM DATA SET INFO:
         DATA SET NAMES IN USE: IXGLOGR.WILLIN.IYLX4.DFHSHUNT.<SEQ#>
         Ext.   <SEQ#>    Lowest Blockid    Highest GMT        Highest Local
        -----  --------  ----------------  -----------------  ---------------
       *00001  A0000000  0000000000001F1E  07/16/97 12:52:22  07/16/97 13:52:

       NUMBER OF DATA SETS IN LOG STREAM: 1
       POSSIBLE ORPHANED LOG STREAM DATA SETS:
         NUMBER OF POSSIBLE ORPHANED LOG STREAM DATA SETS: 0
```

Figure 34. Example output produced by the LIST LOGSTREAM NAME command

If you are using coupling facility log streams, the IXCMIAPU `LIST STRUCTURE NAME(structname) DETAIL(YES)` command is useful in finding the status of CICS log stream structures. For further information about these commands, see z/OS MVS Setting Up a Sysplex.

SMF and RMF statistics

SMF 88 log stream statistics records and RMF coupling facility usage reports are useful for analyzing problems that are affecting performance.

Increasing the amount of coupling facility storage allocated to a structure, or the size of a staging data set, might improve both MVS logger performance and CICS performance.

Obtaining MVS logger and coupling facility dumps

If you suspect there is a problem within the MVS logger which is not a result of some other resolvable problem, you may need to collect additional diagnostic information. The dumps generated by CICS often don't contain sufficient information about the MVS logger.

A dump of XCF and MVS logger address spaces from all systems is useful in the diagnosis of such problems. To obtain the dump, issue the following series of MVS commands:

```
DUMP COMM=(meaningful dump title)
R ww,JOBNAME=(IXGLOGR,XCFAS,cics_jobname),DSPNAME=('IXGLOGR'.*,'XCFAS'.*),CONT
R xx,STRLIST=(STRNAME=structure,(LISTNUM=ALL),ACC=NOLIM),CONT
R yy,REMOTE=(SYSLIST=*('XCFAS','IXGLOGR'),DSPNAME,SDATA),CONT
R zz,SDATA=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,TRT,CSA,GRSQ,XESDATA),END
```

Use the `R xx,STRLIST=(STRNAME=structure,(LISTNUM=ALL),ACC=NOLIM),CONT` instruction only where you suspect a problem with the coupling facility structure.

Error records written to the MVS LOGREC data set may also be useful.

Setting a SLIP trap

The procedure described in the previous section produces snapshots of the MVS logger address space and coupling facility structure at the time that the commands were issued. However, it is usually more useful to take a memory dump when an error occurs.

If you have applied MVS APAR OW27057, a dump of the MVS logger address space is produced automatically if an MVS IXGBRWSE or IXGDELET request fails because the MVS logger cannot find a specific log stream block identifier. (The MVS logger issues a return code of 8 with a reason code of 804.) To cater for other possible logger errors, or to obtain a memory dump of the coupling facility structure associated with a failing log stream, you can set an MVS serviceability level indication processing (SLIP) trap. Setting a SLIP trap causes MVS to take a specified set of actions when a specified event occurs. For example, you could specify that MVS is to take a dump of the MVS logger address space if CICS issues a particular message.

Figure 35 on page 252 shows an example SLIP trap that captures a dump of the CICS address space, the MVS logger address space, and the coupling facility structure associated with the failing logstream.

```
SLIP SET,IF,LPAMOD=(IGC0003E,0),DATA=(1R?+
4,EQ,C4C6C8D3,+8,EQ,C7F0F7F7,+C,EQ,F2),A=S    <change the message
VCD,JOBLIST=(cicsjob,IXGLOGR,XCFAS),          <change CICS Job

-->response xx

xx,DSPNAME=('XCFAS'.*,'IXGLOGR'.*),STRLIST
=(STRNAME=structname,LOCKENTRIES,ACC=NOLIM    <change STRNAME
,(LISTNUM=ALL,

-->response yy

yy,ENTRYDATA=SERIALIZE,ADJUNCT=CAPTURE)),S
DATA=(RGN,XESDATA,ALLNUC,CSA,LSQA,PSA,SQA,
SWA,TRT,COUPLE,WLM,GRSQ,LPA),

-->response zz

zz,ID=LOGR,REMOTE=(JOBLIST,DSPNAME,SDATA),
END
```

Figure 35. An example SLIP trap

In this example, the SLIP triggers when a specific CICS log manager message DFHLG0772 is written to the console. This is specified in the **EQ** parameter of the SLIP:

```
+4,EQ,C4C6C8D3,+8,EQ,C7F0F7F7,+C,EQ,F2)
       D F H L        G 0 7 7        2     <equates to
```

You can also set a more “generic” trap, that is triggered by the occurrence of any one of a range of messages. For example, to cause the SLIP to be triggered by any log manager message in the DFHLG07*xx* range, alter the value of the **EQ** parameter to:

```
+4,EQ,C4C6C8D3,+8,EQ,C7F0F7),
       D F H L        G 0 7      <equates to
```

To use the example SLIP, you must:

1. Replace the `cicsjob` value with the name of the CICS job (or jobs) to be dumped.
2. Replace the `xx`, `yy`, and `zz` values with the appropriate operator reply numbers, as each segment is entered.
3. Replace the `structname` value with the name of the coupling facility structure that contains the failing log stream.

   For system log failures only, you can get the name of the coupling facility structure (or structures) from the two DFHLG0104 messages that were issued when CICS connected to DFHLOG and DFHSHUNT during the run in which the failure occurred.

   For all other log streams, to get the name of the coupling facility structure use the **LIST LOGSTREAM NAME** command already described. For example:

   ```
   //LOGRRPT  EXEC PGM=IXCMIAPU
   //SYSPRINT DD SYSOUT=*
   //SYSIN    DD *
    DATA
      TYPE(LOGR)
      REPORT(YES)
      LIST LOGSTREAM NAME(logstream_name) DETAIL(YES)
   ```

   Figure 34 on page 250 shows example output produced by the **LIST LOGSTREAM NAME** command. Search for the log stream name; the structure name follows it.

   Step “3” on page 252 assumes two things:

- That the failing log stream is a coupling facility log stream. If it is a DASD-only log stream, the **STRLIST** parameter in the example SLIP is not appropriate.
- That the logging problem is repeatable. It is assumed that the log stream has failed at least once before the SLIP is set (the initial failure allowing you to deduce the name of the coupling facility structure to be dumped).

Note:

1. The example SLIP will just fit into the extended operator command area of MVS Version 5 or later.
2. The example SLIP may result in extra dumps being produced for both CICS and the MVS logger address space.

For definitive information about setting SLIP traps, see z/OS MVS Diagnosis: Tools and Service Aids.

Restarting the MVS logger address space

Before you begin

About this task

If the MVS logger address space has failed, you can restart it as follows:

Procedure

1. Use the command `S IXGLOGRS`.

   Note the S at the end. IXGLOGRS restarts IXGLOGR as a system address space.
2. After the MVS logger has restarted, restart all the CICS regions.

Example

What to do next

CAUTION: If you forcibly cancel the MVS logger address space (by issuing a `FORCE IXGLOGR,ARM` command) or coupling facility structures used by the MVS logger (by issuing a `SETXCF FORCE,CON,STRNAME=structname,CONNAME=ALL` command), there is a risk of corruption in the CICS system logs. If the system log is corrupted, CICS issues a message telling you that you need to perform an initial start. Data integrity will be compromised because of the loss of log data required for consistency.

Dealing with a corrupt system log

If the system log becomes corrupt, CICS quiesces. After the system log has been corrupted, it cannot be used again; to get CICS back into production, you must perform an initial start.

About this task

To prevent the problem recurring, you also need to gather diagnosis information that will enable IBM Service to discover why the log was corrupted. Unfortunately, performing an initial start destroys all information from the previous run of CICS. To gather diagnostic information:

Procedure

1. Scan the failed system log, using a utility such as DFHJUP.

   However, the output produced by DFHJUP in these circumstances is not easy to interpret.
2. To supplement DFHJUP's output, perform a diagnostic run of CICS, using the corrupt system log, before performing the initial start.

a) Specify AUTO on the **START** system initialization parameter.

If the system log becomes corrupt, CICS:

- Sets the recovery manager autostart override record in the global catalog so that the next automatic restart of CICS is a diagnostic run (AUTODIAG).
- Issues message DFHRM0152, saying that the next automatic restart will be a diagnostic run, and should be performed before an initial start.

b) If the system log is not corrupt, but you still want to perform a diagnostic run, use the recover manager utility program DFHRMUTL.

For information about DFHRMUTL, see Recovery manager utility (DFHRMUTL).

On a diagnostic run, CICS:

a. Produces a dump of the CICS system state, retrieved from the failed system log.
b. Terminates. Note that, on a diagnostic run, CICS *performs no recovery work and no new work*.

The output produced by a diagnostic run is usually passed to IBM Service.

Benefits of a diagnostic run

The advantages of performing a diagnostic run are:

- It collects diagnostic information automatically, thus allowing you to get CICS back into production quickly.
- When CICS failed, a system dump may not have been produced. A diagnostic run provides one.
- If the diagnostic run is not able to retrieve all the records from the CICS system log, the last record it retrieves shows the point at which the log became unreadable, and may indicate the cause of the problem.
- A diagnostic run allows you to capture a dump of the MVS logger address space. See “Getting dumps of the MVS logger and coupling facility address spaces” on page 254.

Getting dumps of the MVS logger and coupling facility address spaces

For reliable diagnosis, it is important that you have dumps of the MVS logger address space and (if applicable) the coupling facility structures used by the system log.

Before you begin

This means that, before performing the diagnostic run, you will probably need to set a SLIP trap, as described in “Setting a SLIP trap” on page 251.

About this task

You need to set a SLIP if *any* of the following are true:

- You have not applied MVS APAR OW27057.
- MVS did not produce a dump of the logger address space.
- MVS produced a dump of the logger address space but you have not kept the dump.
- The CICS system log uses coupling facility log streams.

When specifying the SLIP, note the following:

Procedure

1. Set the trap for the specific DFHLG07*xx* message that CICS issued when the original failure occurred.

   See the example SLIP in Figure 35 on page 252.

   When the diagnostic run occurs and the failure repeats, the message will drive the SLIP.

Occasionally, the DFHLG07*xx* message that was issued at the time of the original failure is not repeated during a diagnostic run. Instead, a different DFHLG07*xx* message is issued. Therefore the SLIP is not triggered. If this happens, perform another diagnostic run. This time, however, set the SLIP for the DFHLG07*xx* message that was issued during the first diagnostic run.

2. Change the **JOBLIST** parameter in the example SLIP to read

   ```
   JOBLIST=(IXGLOGR,XCFAS),
   ```

   You do not need to specify a dump of the CICS system, because one is taken automatically by the diagnostic run mechanism.

3. Specify a dump of the MVS logger address space.

   See the example SLIP. If you have applied MVS APAR OW27057, and the original failure occurred because the MVS logger was unable to find a specific log stream block identifier, an extra dump may be produced.

4. If the system log uses coupling facility log streams, specify a dump of the coupling facility structure.

   You can get the name of the structure from the two DFHLG0104 messages that were issued when CICS connected to DFHLOG and DFHSHUNT during the run in which the failure occurred.

   If DFHLOG and DFHSHUNT use separate coupling facility structures, dump both structures. Specify the names of both structures on the **STRLIS**T parameter.

Example

Chapter 7. Working with IBM to solve your problem

The IBM Support organization is a global network of centers, with expertise across the IBM product portfolio. It provides you with the responsive software support that you require.

For information about how to contact and work with IBM Support, refer to the IBM Support Guide.

Collecting CICS troubleshooting data (CICS MustGather) for IBM Support

When you encounter a problem in CICS Transaction Server for z/OS (CICS TS), before you contact IBM Support, ensure that you collect troubleshooting data, also known as MustGather. CICS Support can use troubleshooting data to identify and resolve your specific problem. Different symptoms require different data. MustGather documents expedite the troubleshooting process and save you time. The following information applies to all CICS versions unless otherwise specified.

General information

Gather the following general information for every problem record:

1. A complete description of the problem that includes the following questions:
    a. When did the problem first occur?
    b. Is the problem a one time failure or recurring?
    c. Was software or hardware maintenance applied?
    d. Did the failure occur while doing a specific task?
    e. Is the failure occurring in more than one address space?
2. CICS product version, release, and maintenance level
3. Operating system version, release, and maintenance level
4. Related products version, and release levels
5. Your valid contact phone number and email address

Component-specific information

After you collect general information, gather information that is specific to your component. From the following list, click the problem type or component to view a listing of specific documentation that the support team requires to diagnose your problem.

Note: The following links take you to the latest version of CICS TS documentation.

- Abend 878 or 80A
- CICS-MQ adapter or CICS-MQ bridge
- CICSPlex® SM
- CMCI JVM server
- Coupling facility data table server
- Db2®
- Dynamic Scripting for CICS TS 5.1 or later
- Dynamic Scripting for CICS TS 4.2 or CICS TS 4.1
- Event processing
- Explorer errors or incorrect output
- Explorer UI not available or not stable

- File control (non-RLS)
- File control (RLS)
- IMS Database Control (DBCTL)
- IBM Documentation
- Intersystem communication (ISC)
- IP interconnectivity (IPIC)
- Java (JVM server)
- Logger
- Language Environment (LE) abend
- LIBRARY management
- Mobile extensions
- Multiregion operation (MRO)
- Node.js applications
- Performance
- Policies
- Program checks or abends
- SFR (Service Flow Runtime)
- Shared data tables (SDT)
- Short on storage (SOS)
- Storage violation
- Terminal hang
- Temporary storage servers
- Waits or loops
- Web services, XML, and JSON transformation

Tip:

- Search the CICS support site for known problems using symptoms like the message number and error codes.
- If you find a fixing PTF, see Ordering CICS products and maintenance for the options that are available to order CICS maintenance.
- Gather the documentation and work with the CICS support team to resolve your problem.

The global trap exit DFHTRAP

The global trap exit DFHTRAP is an Assembler language program that can be invoked when the CICS trace domain is called to write a trace entry. DFHTRAP is intended to be used only under the guidance of IBM Service personnel, to make a detailed diagnosis of a problem without having to stop and restart CICS.

Typically, the global trap exit is used to detect errors that cannot be diagnosed by other methods:

- Errors that occurred some time before the effects are noticed
- Errors that cause intermittent problems that are difficult to reproduce

For example, a field might be changed to a bad value, or some structure in storage might be overlaid at a specific offset.

A skeleton version of DFHTRAP is supplied in both source and load-module forms. The source of DFHTRAP is cataloged in the CICS710.SDFHMAC library.

Installing and controlling the DFHTRAP exit

Before you use the global trap exit DFHTRAP you must install and activate it.

About this task

You can install and activate the global trap exit DFHTRAP at CICS initialization or while CICS is running. You can also deactivate the exit while CICS is running.

If a program check occurs in DFHTRAP, the recovery routine in DFHTRPT marks the exit as unusable, and it is ignored on future invocations of the trace domain. CICS issues the message DFHTR1001 to the system console and takes a CICS system dump with dump code TR1001, showing the PSW and registers at the time of the interrupt. To recover from this situation, you must replace the current version of DFHTRAP.

Procedure

- Install DFHTRAP as part of the RDO group DFHFE, using one of the following methods:
  - The **GRPLIST** system initialization parameter.
  - The CEDA transaction while CICS is running.
- To activate the global trap exit at CICS initialization, specify the system initialization parameter **TRAP=ON**, either in DFHSIT or as a startup override.
- To activate or reactivate the global trap exit while CICS is running, issue the following command:

```
CSFE DEBUG,TRAP=ON
```

- To deactivate the global trap exit, issue the following command:

```
CSFE DEBUG,TRAP=OFF
```

- To replace the current version of DFHTRAP, issue the following sequence of commands:

```
CSFE DEBUG,TRAP=OFF
CEMT SET PROGRAM(DFHTRAP) NEWCOPY
CSFE DEBUG,TRAP=ON
```

Related tasks

Coding the DFHTRAP exit
The source of the skeleton version of the global trap exit DFHTRAP contains comments that explain its use of registers and DSECTs, and the coding required to use the exit.

Related reference

Information passed to the DFHTRAP exit
The CICS trace domain passes information to the DFHTRAP exit in a parameter list addressed by register 1. The DSECT DFHTRADS is supplied for this list.

Actions the DFHTRAP exit can take
The global trap exit can set a return-action flag byte to tell the trace domain what action is required on return from the exit.

Information passed to the DFHTRAP exit

The CICS trace domain passes information to the DFHTRAP exit in a parameter list addressed by register 1. The DSECT DFHTRADS is supplied for this list.

DFHTRADS contains the addresses of the following items:

- The return-action flag byte
- The trace entry that has just been added to the trace table

- Up to three data fields for data below the bar or above the bar (in 64-bit storage) to be included in a further trace entry
- An 80-byte work area for the sole use of the global trap exit
- The CICS common system area (CSA)
- The task control area (TCA), if there is one
- A register save area

The CSA address is zero for invocations of DFHTRAP early in initialization, before the CSA is acquired.

The DSECT also contains EQU statements for use in setting the return-action flag byte.

The global trap 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 for DFHTRAP.

Related tasks

Installing and controlling the DFHTRAP exit
Before you use the global trap exit DFHTRAP you must install and activate it.

Coding the DFHTRAP exit
The source of the skeleton version of the global trap exit DFHTRAP contains comments that explain its use of registers and DSECTs, and the coding required to use the exit.

Related reference

Actions the DFHTRAP exit can take
The global trap exit can set a return-action flag byte to tell the trace domain what action is required on return from the exit.

Actions the DFHTRAP exit can take

The global trap exit can set a return-action flag byte to tell the trace domain what action is required on return from the exit.

Any combination of the following actions can be specified:

- Do nothing.
- Make a further trace entry using data in 24 bit storage or 31 bit storage (below the bar).
- Make a further trace entry using data in 64 bit storage (above the bar).
- Take a CICS system dump. There are two possible actions here, take a system dump while holding the trace lock using system dump code TR1004, or take a system dump without holding the trace lock using system dump code TR1003. Holding the trace lock while the system dump is taken is more invasive to the system but might be required if debugging a concurrency problem.
- Terminate CICS without a system dump after message DFHTR1000. (If you require a system dump, the system dump return-action flag must also be set.)
- Disable the trap exit, so that it is not invoked again until you issue the command CSFE `DEBUG,TRAP=ON`.

All actions are honored on return to the trace domain.

The skeleton source for DFHTRAP shows how to make a further trace entry. When DFHTRAP detects a TS GET request, you can ask for a further trace entry to be made by entering the data required in the data fields supplied for this purpose, and by setting the appropriate bit in the return-action flag byte. If the required data is in 64 bit storage (above the bar), you must use the 64 bit versions of the data fields, and set the appropriate bit in the return-action flag byte to show that 64 bit data is used. The trace domain makes a trace entry with trace point ID TR 0103, incorporating the information supplied by the exit.

Trace entries created following a request by DFHTRAP are written to the currently active trace destination. This could be the internal trace table, the auxiliary trace data set, or the GTF trace data set.

The skeleton source for DFHTRAP also shows how to detect the trace entry made by the storage manager (SM) domain for a GETMAIN request for a particular subpool. The skeleton source shows you how to look at the data fields within a trace entry.

Related tasks

Installing and controlling the DFHTRAP exit
Before you use the global trap exit DFHTRAP you must install and activate it.

Coding the DFHTRAP exit
The source of the skeleton version of the global trap exit DFHTRAP contains comments that explain its use of registers and DSECTs, and the coding required to use the exit.

Related reference

Information passed to the DFHTRAP exit
The CICS trace domain passes information to the DFHTRAP exit in a parameter list addressed by register 1. The DSECT DFHTRADS is supplied for this list.

Coding the DFHTRAP exit

The source of the skeleton version of the global trap exit DFHTRAP contains comments that explain its use of registers and DSECTs, and the coding required to use the exit.

About this task

The skeleton version of DFHTRAP is supplied in both source and load-module forms. The source of DFHTRAP is cataloged in the CICS710.SDFHMAC library. DFHTRAP runs in AMODE(64).

The 80-byte work area that is provided for the sole use of the global trap exit is in 64-bit storage (storage above the bar). When the global trap exit is activated, the trace domain acquires the storage and initializes it to binary zeros. The working storage exists until the global trap exit is deactivated by the command CSFE DEBUG,TRAP=OFF. In a dump, the DFHTRAP working storage is located as follows:

- In a CICS transaction dump, the information about DFHTRAP is 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 39 for details of how to use the TR keyword to format the trace domain information in the dump.

For information about the actions that you can specify for DFHTRAP, see “Actions the DFHTRAP exit can take” on page 260.

Procedure

- Ensure that the code in DFHTRAP does not use any CICS services, cause the current task to lose control, or change the status of the CICS system.
- Ensure that DFHTRAP saves and restores 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.
- Ensure that DFHTRAP is specified as AMODE(64) and RMODE(ANY).

  DFHTRAP might switch addressing mode while it is running, but it must always return control to the trace domain in 64-bit mode.
- Ensure that the library search sequence in the CICS startup JCL finds the correct version of the load module.

Related tasks

Installing and controlling the DFHTRAP exit

Before you use the global trap exit DFHTRAP you must install and activate it.

Related reference

Information passed to the DFHTRAP exit

The CICS trace domain passes information to the DFHTRAP exit in a parameter list addressed by register 1. The DSECT DFHTRADS is supplied for this list.

Actions the DFHTRAP exit can take

The global trap exit can set a return-action flag byte to tell the trace domain what action is required on return from the exit.

Chapter 8. SDUMP contents and IPCS CICS VERBEXIT keywords

This section provides a cross-reference between the CICS control blocks contained in an SDUMP and their associated IPCS CICS VERBEXIT keyword.

Keyword to control block map

This reference provides a list of IPCS CICS VERBEXIT keywords and the CICS control blocks that they display.

AI keyword

- AITMSSA (AITM static storage)
- AITMTE (AITM entry)

AP keyword

- CICS24 (task storage, below 16 MB, CICS key)
- CICS31 (task storage, above 16 MB, CICS key)
- CICS64 (task storage, above the bar, CICS key)
- 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)
- SYSEIB (system EXEC interface block)
- SYS_TCA (task control area, system area only)
- TCA (task control area, user)
- TD (user transient data)
- TS (user temporary storage)
- USER24 (task storage, below 16 MB, user key)
- USER31 (task storage, above 16 MB, user key)
- USER64 (task storage, above the bar, user key)

APS keyword

- CICS24 (task storage, below 16 MB, CICS key)
- CICS31 (task storage, above 16 MB, CICS key)
- 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)

- SYSEIB (system EXEC interface block)
- SYS_TCA (task control area, system area only)
- TCA (task control area, user)
- TD (user transient data)
- TS (user temporary storage)
- USER24 (task storage, below 16 MB, user key)
- USER31 (task storage, above 16 MB, user key)

Refer to the z/OS Language Environment Debugging Guide for details of LE control blocks and data areas.

AS keyword

- ASA (asynchronous services domain anchor block)
- ACCB (asynchronous coordination control block)
- ACTXN (Asynchronous Parent Transaction Token)

BR keyword

- Bridge facility bitmap
- Bridge facility block
- Bridge facility keep chain
- BRXA (bridge exit interface)
- TXN_CS

CP keyword

- CPSTATIC (common programming interface static storage)

CSA keyword

- CSA (common system area)
- CSAOPFL (CSA optional features list)
- CWA (common work area)

DB2 keyword

- D2CSB (CICS DB2 subtask block)
- D2ENT (CICS DB2entry control block)
- D2GLB (CICS DB2 global block)
- D2LOT (CICS DB2 life of task block)
- D2PKGSET (CICS DB2 PACKAGESET control block)
- D2SS (CICS DB2 static storage)
- D2TRN (CICS DB2tran control block)

DD keyword

- ANCHOR (directory manager anchor block)
- AVL_HEAD (AVL tree header)
- BRWS_VAL (browse value)
- DIR_HEAD (directory header)
- HASH_TBL (hash table)
- HASHELEM (collision list element)

DH keyword

- DBB (document bookmark block)

- DCR (document control record)
- DDB (document data block)
- DHA (document handler domain anchor block)
- DOA (document anchor block)

DLI keyword

- CWE (CICS-DBCTL control work element)
- DFHDLP (CICS-DLI interface parameter list)
- DGB (CICS-DBCTL global block)
- DGBCTA (DBCTL transaction area)
- DSB (CICS-DBCTL scheduling block)
- DXPS (CICS-DL/I-XRF anchor block)
- PAPL (DL/I-DRA architected parameter list)
- RSB (remote scheduling block)
- SYS_TCA (TCA system area only)
- TCA (task control area)

DM keyword

- DMANCHOR (domain manager anchor block)
- WQP (domain wait queue)

DP keyword

- DPA (DP domain anchor block)
- DPTA (DP domain task area)

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)

EC keyword

- DFHECSS (Event capture static storage)
- DFQE (Deferred filter queue element)
- ECAF (Failed EP adapter in EP adapter set)
- ECCD (Event capture application event capture data item)
- ECCS (Event capture specification)
- ECEVB (Event binding)
- ECFP (Event capture application event filter predicate)
- ECFPX (Event capture application event filter predicate extension)

- ECSCD (Event capture system event capture data item)
- ECSFP (Event capture system event filter predicate)

EJ keyword

- DFHEJANC (EJ domain anchor block)
- DFHEJANE (EJ domain elements anchor area)
- EJAO (EJ domain object store anchor area)
- EJDU_BLOCK (Task related Java diagnostics)
- OS_ELEMENT (EJ object store elements)
- SYSLIB (DFHEJDUB)

EM keyword

- EMA (EM anchor block)
- EVA (event pool anchor)
- EVB (event pool block)

EP Keyword

- ECQE (Event processing queue element)
- EDTB (Event processing dispatcher task block)
- EPA (Event processing anchor block)
- EPAC (Event processing adapter configuration data)
- EPADA (Event processing adapter)
- EPADI (EP adapter name in EP adapter set)
- EPADT (EP adapter set)

FCP keyword

- ACB (VSAM ACBs)
- AFCTE (application file control table element)
- DCB (data control block)
- DFHDTTABLE (data table base area)
- DFHDTFILE (data table path area)
- DFHDTHEADER (data table global area)
- DSNB (data set name block)
- DTRGLOBL (data table remote global area)
- FBWA (data table browse area)
- FCSTATIC (FCP static storage, anchor block)
- FCTE (file control table element)
- FLAB (file lasting access block)
- FRAB (file request anchor block)
- FRTE (file request thread element)
- SHRCTL (shared LSRPOOLs)
- VSWA (VSAM work area)

ICP keyword

- ICE (interval control elements/AIDs)

IE keyword

- Control blocks associated with TCP/IP conversations.

IS keyword
IPCONN control blocks

KE keyword

See note 1.

- AFCB (CICS AFCB)
- AFCS (CICS AFCS)
- AFT (CICS AFT)
- AUTOSTCK (automatic storage stack entry)
- DOH (domain table header)
- DOM (domain table entry)
- KCB (kernel anchor block)
- KERNSTCK (kernel linkage storage stack entry)
- KERRD (kernel error data)
- KTCB (KTCB table entry)
- TAH (task table header)
- TAS (task table entry, TASENTRY)
- TCH (KTCB table header)

LD keyword

See note 1.

- 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)

LG keyword

- LGA (log domain anchor)
- LGBR (stream, journal, journalmodel browse)
- LGGL (general log data)
- LGJI (journal information)
- LGJMC (journalmodel content)
- LGSD (stream data)
- LGUOW (log manager unit of work token)
- STATSBUF (log manager statistics)
- Block class data
- Block instance data
- BrowseableStream class data
- BrowseableStream instance data
- Chain class data
- Chain instance data
- HardStream instance data
- L2 anchor block

- Stream class data
- Stream instance data
- SuspendQueue elements
- SystemLog class data

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)

ML keyword

- MLA (markup language 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)
- MNEXC (exception record buffer)
- MNEXLIST (user EMP address list)
- MNFLDMAP (excluded/included CICS field map)
- MNPER (performance data buffer)
- MNSMF (SMF record buffer)
- MNTMA (transaction monitoring area)
- MNWLMPB (MVS WLM performance blocks)

MP keyword

- MPA (MP domain anchor block)
- MPMOD (MP model)
- MPMODR (MP model rule)
- MPPFA (MP policy event action failed EP adapter in EP adapter set block)
- MPPMB (MP policy modifier block)
- MPPPB (MP policy block)
- MPPRB (MP policy rule block)
- MPTAS (MP task storage)

MRO keyword

See note 1.

- CCB (connection control block)

- CRB (CICS region block)
- CSB (connection status block)
- 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)

OT keyword

- OTAN (OT domain anchor block)

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)
- SITDLI (SIT DL/I extension)
- TOGGLE (feature toggle list)

PCP keyword

- PPTTE (program processing table entries - program resource definitions)

PCT keyword

- TXD64 (transaction definition instance 64 bit extension)
- TXDINST (transaction definition instance)
- TXDSTAT (transaction definition static data)

PG keyword

- CHCB (channel control block)
- CPCB (container pool control block)
- CRCB (container control block)
- CSCB (container segment block)
- HTB (handle table)
- LLE (load list element, can be system LLE or task LLE)
- PGA (program manager anchor)
- PGWE (program manager wait element)
- PLCB (program manager program level control block)
- PPTE (program processing table element)
- PTA (program transaction area)

For an explanation of PG summary data in a level-1 dump, see “Summary data for PG and US keywords” on page 287.

PI keyword

- HPE (header program element)
- PEB (pipeline element block)
- PIA (pipeline manager anchor block)
- PIH (pipeline element header block)
- SNE (service handler element)
- WCB (Web service control block)
- WHB (Web service header block)
- WRB (Web service resource block)

RL keyword

- RLA (resource life-cycle domain anchor block)

RM keyword

- RMCD (recovery manager client directory)
- RMCI (recovery manager client identity)
- RMDM (recovery manager domain anchor)
- RMLI (recovery manager loggable identity)
- RMLK (recovery manager link)
- RMNM (recovery manager logname)
- RMRO (recovery manager resource owner)
- RMSL (recovery manager system log)
- RMUW (recovery manager)

RS keyword

- RSA (Region status domain anchor block)

RX keyword

- Active Unit of Recovery data (CICS key)
- Active Unit of Recovery data (Key 0)
- DFHRXSVC dynamic storage area
- In-resync Unit of Recovery data (CICS key)
- In-resync Unit of Recovery data (Key 0)
- RX domain anchor block (CICS key)
- RX domain anchor block (Key 0)

RZ keyword

- RZDM (RZ domain anchor block)
- RZREQSTR (rz_reqstream instance data)
- RZRMB (rzrmb instance data)
- RZTR (rztr instance data)

SJ keyword

- SJA (JVM domain anchor block)
- SJCCH (Shared class cache control block)
- SJTCB (JVM TCB control block)
- SJVMS (JVMset or control block)

SM keyword

- CTN (cartesian tree node)
- DXE (DSA extent list element)
- DXG (DSA extent getmain description)
- DXH (DSA extent list header)
- GPAM (page allocation map for 64 bit DSA)
- GPPA (page pool control area for 64 bit DSA)
- GPPX (page pool extent control area for 64 bit DSA)
- 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)
- SUA (subspace area)

SO keyword

- LTE (Listener table entry)
- SOA (SO domain anchor block)
- STE (Session table entry)
- TDA (Tcpipservice anchor block)
- TDB (Tcpipservice control block)
- TBR (Tcpipservice browse block)

SSA keyword

- SSA (static storage areas)
- SSAL (static storage address list)

ST keyword

- STANCHOR (statistics domain anchor block)
- STSAFPB (statistics authorization facility parameter block)
- STSMF (statistics SMF record)
- STSTATS (statistics domain statistics record)

SZ keyword

- SZSDS (FEPI static area)

TCP keyword

- ACB (z/OS Communications Server access method control block)
- AID (automatic initiation descriptor)
- AWE (autoinstall work element)
- BIND (bind image)
- BITMAPn (one of several resource naming BITMAPs)
- CCIN (CICS client CCIN parameters)
- CTIN (CICS client CTIN parameters)
- DCB (BSAM data control block)
- DIB (data interchange block)
- DUMTCTTE (dummy TCTTE)
- EXLST (z/OS Communications Server ACB exit list)
- ISORM (indirect system object resolution map)
- LOGDS (extracted logon or CLSDST Pass data)
- LUITE (local userid table element)
- NIB (node initialization block)
- NIBLIST (persistent sessions INQUIRE PERSESS list of NIBs)
- PRSS (persistent sessions CV29, FMH5, BIS, and BID data)
- PS_BID (persistent sessions OPNDST RESTORE BID)
- PS_BIND (persistent sessions INQUIRE PERSESS BIND)
- PS_BIS (persistent sessions OPNDST RESTORE BIS)
- PS_CV29 (persistent sessions OPNDST RESTORE control vector 29)
- PS_FMH5 (persistent sessions OPNDST RESTORE FMH5)
- PS_MODN (persistent sessions INQUIRE PERSESS modename)
- PS_NIB (persistent sessions INQUIRE PERSESS NIB)
- PS_PLST (persistent sessions INQUIRE PERSESS parameter list)
- PS_POOL (persistent sessions RPL pool header)
- PS_RPL (persistent sessions RPL)
- PS_SESS (persistent sessions INQUIRE PERSESS session ID)
- PWE (postponed work element)
- RACE (receive-any control elements)
- RAIA (z/OS Communications Server receive-any Input area)
- RPL (z/OS Communications Server request parameter list, receive-any RPLs)
- SNEX (TCTTE signon extension)
- TACLE (terminal abnormal condition line entry)
- TCTENIB (NIB descriptor)
- TCTESBA (APPC send/receive buffer)
- TCTFX (TCT prefix)
- TCTLE (TCT line entries)
- TCTTELUC (TCTTE APPC extension)
- TCTME (TCT mode entry)
- TCTSE (TCT system entries)
- TCTSK (TCT skeleton entries)

- TCTTE (TCT terminal entries)
- TCTTECCE (console control element)
- 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 (transient data queue definitions)
- 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)
- TDCUB (TD CI update block)
- TDST (TD static storage)
- TDQUB (TD queue update block)
- TDUA (TD UOW anchor block)
- VEMA (TD VSAM error message area)

TI keyword

- TIA (Timer domain anchor block)
- TRE (Timer request elements)

TK keyword

- EIB (EXEC interface block)
- EIUS (EXEC interface user structure)
- KTCB (KTCB table entry)
- PLCB (Program level control block)
- SYSEIB (system EXEC interface block)
- SYS_TCA (task control area, system area only)
- TCA (task control area, user)
- TMA (Transaction monitoring area)
- TXN (Transaction control block)

TKS keyword

- EIB (EXEC interface block)
- EIUS (EXEC interface user structure)
- KTCB (KTCB table entry)

- PLCB (Program level control block)
- SYSEIB (system EXEC interface block)
- SYS_TCA (task control area, system area only)
- TCA (task control area, user)
- TMA (Transaction monitoring area)
- TXN (Transaction control block)

TMP keyword

- DIRSEG (directory segments)
- SKT (scatter tables)
- TM_LOCKS (read lock blocks)
- TMSTATIC (table manager static storage)

TR keyword

See note 1.

- TRA (trace domain anchor block)
- TRDCB (auxiliary trace data set DCB)
- TRDECB (auxiliary trace data set DECB)

TS keyword

- ACA (TS auxiliary control area)
- BCA (TS buffer control area)
- BMH (TS byte map header)
- BMP (TS byte map)
- BRB (TS browse block)
- DTN (TS digital tree node)
- ICE (interval control element)
- PCA (TS pool control area)
- QAB (TS queue anchor block)
- QOB (TS queue ownership block)
- QUB (TS queue update block)
- SBB (TS shared browse block)
- STE (TS sysid table entry)
- TSA (TS anchor block)
- TSBUFFER (TS I/O buffer)
- TSI (TS item descriptor)
- TSM (TS main item header)
- TSHANCH (TS shared class anchor)
- TSMNANCH (TS main class anchor)
- TSMODEL (TS model class anchor)
- TSNANCH (TS name class anchor)
- TSOANCH (TS ownership lock class anchor)
- TSQ (TS queue control block)
- TSQANCH (TS queue class anchor)
- TSS (TS aux section descriptor)

- TST (TS table header)
- TSTTE (TS table entry)
- TSW (TS wait element)
- TSX (TS aux item descriptor)
- VCA (TS VSWA control area)
- VSWA (TS VSAM work area)
- XRH (TS aux record header)

UEH keyword

- EPB (exit program blocks)
- GWA (EPB global work area)
- TIE (task interface element)
- UET (user exit table)

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 "Summary data for PG and US keywords" on page 287.

WB keyword

- GWA (Global work area)
- WBABC (Web anchor block)
- WBSTC (Web state manager blocks)

XM keyword

- MXT (XM domain MXT tclass)
- TCL (XM domain tclass)
- TXN (XM domain transaction)
- XM_XB (XM domain browse element)
- XMA (XM domain anchor block)

XRF keyword

- CAVM_STA (CAVM static storage)
- XRP_ACTS (XRP active status area)
- XRP_ALTS (XRP alternate status area)
- XRP_HLTH (XRP health area)
- XRP_XRSA (XRP anchor area)
- XRPSTAT (XRP static storage)

XS keyword

- XSA (security domain anchor block)
- XSSS (security supervisor storage)

Note 1: The keyword can also be used when formatting an EXCI SDUMP, that is a dump of a non-CICS address space that is using EXCI to communicate with CICS. See Formatting system dumps.

Control block to keyword map

This reference provides a list of all CICS control blocks in an SDUMP, alphabetically, with their associated IPCS CICS VERBEXIT keyword.

CICS control block		VERBEXIT keyword
ACA	TS auxiliary control area	TS
ACB	VSAM ACBs	FCP
ACB	z/OS Communications Server 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
ANCHOR	Directory manager anchor block	DD
APE	Active program element	LD
ASA	Asynchronous services domain anchor block	AS
AUTOSTCK	Automatic storage stack entry	KE
AVL_HEAD	AVL tree header	DD
AWE	Autoinstall work element	TCP
BCA	TS buffer control area	TS
BIND	Bind image	TCP
BITMAPn	Instance of resource naming BITMAP	TCP
BMH	TS byte map header	TS
BMP	TS byte map	TS
BRB	TS browse block	TS
BRWS_VAL	Browse value	DD
BRXA	Bridge exit interface	BR
	Bridge facility bitmap	BR
	Bridge facility keep chain	BR
	Bridge facility block	BR
	TXN_CS	BR
BUFFER	TD I/O buffer	TDP

CICS control block		VERBEXIT keyword
CAVM_STA	CAVM static storage	XRF
CC_ACB	Local catalog ACB	CC
CC_RPL	Local catalog RPLs, one each thread	CC
CCB	Connection control block	MRO
CCBUFFER	Local catalog buffers, one each thread	CC
CCIN	CICS client CCIN parameters	TCP
CHCB	Channel control block	PG
CICS24	Task storage - below 16 MB, CICS key	AP APS
CICS31	Task storage - above 16 MB, CICS key	AP APS
CICS64	Task storage - above the bar, CICS key	AP APS
CPCB	Container pool control block	PG
CPE	Current program element	LD
CPSTATIC	Common program interface storage	CP
CRB	CICS region block	MRO
CRCB	Container control block	PG
CSA	Common system area	CSA
CSCB	Container segment block	PG
CSAOPFL	CSA optional features list	CSA
CSB	Connection status block	MRO
CSECTL	Program CSECT list	LD
CTIN	CICS client CTIN parameters	TCP
CTN	Cartesian tree node	SM
CWA	Common work area	CSA
CWE	CICS/DBCTL control work element	DLI
D2CSB	CICS DB2 subtask block	DB2
D2ENT	CICS DB2entry control block	DB2
D2GLB	CICS DB2 global block	DB2
D2LOT	CICS DB2 life of task block	DB2
D2PKGSET	CICS DB2 PACKAGESET control block	DB2
D2SS	CICS DB2 static storage	DB2
D2TRN	CICS DB2tran control block	DB2
DCB	Data control block	FCP
DCB	BSAM data control block	TCP
DCTE	Destination control table entries (transient data queue definitions)	TDP

CICS control block		VERBEXIT keyword
DFHDLP	CICS/DLI interface parameter list	DLI
DFHDTTABLE	Data table base area	FCP
DFHDTFILE	Data table path area	FCP
DFHDTHEADER	Data table global area	FCP
DFHECSS	Event capture static storage	EC
DFQE	Deferred filter queue element	EC
DFHEJANC	EJ domain anchor block	EJ
DFHEJANE	EJ domain elements anchor area	EJ
DFHSIT	System initialization table	PA
DGB	CICS/DBCTL global block	DLI
DGBCTA	DBCTL transaction area	DLI
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
DPA	DP domain anchor block	DP
DPTA	DP domain task area	DP
DS_TCB	TCB block	DS
DSANC	Dispatcher anchor block	DS
DSB	CICS/DBCTL scheduling block	DLI
DSNB	Data set name block	FCP
DTA	Dispatcher task area	DS
DTN	TS digital tree node	TS
DTRGLOBL	Data table remote global area	FCP
DUA	Dump domain anchor block	DU
DUBUFFER	Transaction dump-data set buffer	DU
DUMTCTTE	Dummy TCTTE	TCP
DWE	User DWE storage	AP APS
DXE	DSA extent list element	SM
DXG	DSA extent getmain descriptor	SM
DXH	DSA extent list header	SM
DXPS	CICS-DL/I-XRF anchor block	DLI
ECAF	Failed EP adapter in EP adapter set	EC

CICS control block		VERBEXIT keyword
ECCD	Event capture application event capture data item	EC
ECCS	Event capture specification	EC
ECEVB	Event binding	EC
ECFP	Event capture application event filter predicate	EC
ECFPX	Event capture application event filter predicate extension	EC
ECQE	Event processing queue element	EP
ECSCD	Event capture system event capture data item	EC
ECSFP	Event capture system event filter predicate	EC
EDTB	Event processing dispatcher task block	EP
EIB	EXEC interface block	AP APS
EIS	EXEC interface structure	AP APS
EIUS	EXEC interface user structure	AP APS
EJAO	EJ domain object store anchor area	EJ
EJDU_BLOCK	Task-related Java diagnostics	EJ
EPA	Event processing anchor block	EP
EPAC	Event processing adapter configuration data	EP
EPADA	Event processing adapter	EP
EPADI	EP adapter name in EP adapter set	EP
EPADT	EP adapter set	EP
EPB	Exit program blocks	UEH
EXLST	z/OS Communications Server 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 APS
FLAB	File lasting access block	FCP
FRAB	File request anchor block	FCP
FREECHAI	LM domain freechain 1	LM
FREECHAI	LM domain freechain 2	LM
FREECHAI	LM domain freechain 3	LM
FRTE	File request thread element	FCP
GC_ACB	Global catalog ACB	CC
GC_RPL	Global catalog RPLs, one each thread	CC
GCBUFFER	Global catalog buffers, one each thread	CC
GPAM	Page allocation map for 64 bit DSA	SM

CICS control block		VERBEXIT keyword
GPPA	Pagepool control area for 64 bit DSA	SM
GPPX	Pagepool extent control area for 64 bits DSA	SM
GWA	EPB global work area	UEH
GWA	Web global work area	WB
HASH_TBL	Hash table	DD
HASHELEM	Collision list element	DD
HPE	Header program element	PI
HTB	Handle table	PG
ICE	Interval control elements/AIDs	ICP
ICE	TS interval control element	TS
IECCB	IE domain client conversation block	IE
IECSB	IE domain connection status block	IE
ISORM	Indirect system object resolution map	TCP
JCA	Journal control area	AP APS
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
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
LGA	Log domain anchor block	LG
LGBR	Stream/journal/journalmodel browse	LG
LGGL	General log data	LG
LGJI	Journal information	LG
LGJMC	Journalmodel content	LG
LGSD	Stream data	LG
LGUOW	Log manager unit of work token	LG
	Block class data	LG
	Block instance data	LG
	BrowseableStream class data	LG
	BrowseableStream instance data	LG
	Chain class data	LG

CICS control block		VERBEXIT keyword
	Chain instance data	LG
	HardStream instance data	LG
	L2 anchor block	LG
	Stream class data	LG
	Stream instance data	LG
	SuspendQueue elements	LG
	SystemLog class data	LG
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
LUITE	Local userid table element	TCP
LXA	LX array	MRO
MAPCOPY	User BMS MAP storage	AP APS
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
MLA	Markup language domain anchor block	ML
MNA	Monitor domain global storage - anchor block	MN
MNAFB	Monitor authorization facility parameter block	MN
MNCONNS	Monitor field connectors	MN
MNDICT	Monitor dictionary	MN
MNEXC	Exception record buffer	MN
MNEXLIST	User EMP address list	MN
MNFLDMAP	Excluded/included CICS field map	MN
MNPER	Performance data buffer	MN
MNSMF	SMF record buffer	MN
MNTMA	Transaction monitoring area	MN
MNWLMPB	MVS WLM performance blocks	MN
MPA	MP domain anchor block	MP

CICS control block		VERBEXIT keyword
MPMOD	MP model	MP
MPMODR	MP model rule	MP
MPPFA	MP policy event action failed EP adapter in EP adapter set block	MP
MPPMB	MP policy modifier control block	MP
MPPPB	MP policy block	MP
MPPRB	MP policy rule block	MP
MPTAS	MP task storage	MP
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
NIBLIST	Persistent session INQUIRE PERSESS NIBs	TCP
OPENBLOK	Transaction dump open block	DU
OS_ELEMENT	EJ object store elements	EJ
OTAN	OT domain anchor block	OT
OVERSTOR	Override parameter temporary work area	PA
PAA	Parameter manager domain anchor block	PA
PAM	Page allocation map	SM
PAPL	DLI/DRA architected parameter list	DLI
PARMSAVE	SIT override parameters	PA
PCA	TS pool control area	TS
PEB	Pipeline element block	PI
PGA	Program management anchor	PG
PGWE	Program management wait element	PG
PIA	Pipeline manager anchor block	PI
PIH	Pipeline element header block	PI
PLCB	Program management program control block	PG
PPA	Pagepool control area	SM
PPTE	Program processing table element	PG
PPTTE	Program processing table entries (program definitions)	PCP
PPX	Pagepool extent control area	SM
PRSS	Persistent sessions CV29, FMH5, BIS, and BID data	TCP

CICS control block		VERBEXIT keyword
PRSTATIC	Partner resource static area	PR
PRTE	Partner resource table entries	PR
PRVMODS	SIT PRVMOD list	PA
PS_BID	Persistent sessions OPNDST RESTORE BID	TCP
PS_BIND	Persistent sessions INQUIRE PERSESS BIND	TCP
PS_BIS	Persistent sessions OPNDST RESTORE BIS	TCP
PS_CV29	Persistent sessions OPNDST RESTORE control vector 29	TCP
PS_FMH5	Persistent sessions OPNDST RESTORE FMH5	TCP
PS_MODN	Persistent sessions INQUIRE PERSESS modename	TCP
PS_NIB	Persistent sessions INQUIRE PERSESS NIB	TCP
PS_PLST	Persistent sessions INQUIRE PERSESS parameter list	TCP
PS_POOL	Persistent sessions RPL pool header	TCP
PS_RPL	Persistent sessions RPL	TCP
PS_SESS	Persistent sessions INQUIRE PERSESS session ID	TCP
PTA	Program transaction area	PG
PWE	Postponed work element	TCP
QAB	TS queue anchor block	TS
QOB	TS queue ownership block	TS
QPH	Quickcell page header	SM
QPK	Quickcell page free element	SM
QUB	TS queue update block	TS
RACE	Receive-any control elements	TCP
RAIA	z/OS Communications Server receive-any input area	TCP
RMCBS	Recovery manager control blocks	RM
RPL	z/OS Communications Server request parameter list - receive-any RPLs	TCP
RPL	TD VSAM RPL	TDP
RSA	Region status domain anchor block	RS
RSB	Remote scheduling block	DLI
RZDM	RZ domain anchor block	RZ
RZREQSTR	rz_reqstream instance data	RZ
RZRMB	rzrmb instance data	RZ
RZTR	rztr instance data	RZ
SAE	Storage access table entry	SM
SAT	Storage access table	SM

CICS control block		VERBEXIT keyword
SBB	TS shared browse block	TS
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	Shared LSRPOOLs	FCP
SITDLI	SIT DL/I extension	PA
SJA	JVM domain anchor block	SJ
SJCCH	Shared class cache control block	SJ
SJTCB	JVM TCB control block	SJ
SJVMS	JVMset or control block	SJ
SKT	Scatter tables	TMP
SLCB	Subsystem logon control block	MRO
SMA	Storage manager domain anchor block	SM
SMX	Transaction storage area	SM
SNE	Service handler element	PI
SNEX	TCTTE signon extension	TCP
SQE	Suspend queue element	SM
SSA	Static storage areas	SSA
SSAL	Static storage address list	SSA
STANCHOR	Statistics domain anchor block	ST
STATSBUF	Log manager statistics	LG
STE	TS sysid table entry	TS
STSAFPB	Statistics authorization facility parameter block	ST
STSMF	Statistics SMF record	ST
STSTATS	Statistics domain statistics record	ST
SUA	Subspace area	SM
SUDB	Subsystem user definition block	MRO
SUSPAREA	Unformatted SUSPEND_AREAS/TOKENS	DS
SYSLIB	DFHEJDUB	EJ
SYS_TCA	TCA - system	DLI AP

CICS control block		VERBEXIT keyword
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	DLI 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
TCTLE	TCT line entries	TCP
TCTME	TCT mode entry	TCP
TCTSE	TCT system entries	TCP
TCTSK	TCT skeleton entries	TCP
TCTTE	Terminal control table terminal entries (terminal definitions)	TCP
TCTTECCE	Console control element	TCP
TCTTELUC	TCTTE LU6.2 extension	TCP
TCTTETTE	TCTTE extension	TCP
TCTTEUA	TCTTE user area	TCP
TD	User transient data	AP APS
TDCUB	TD CI update block	TDP
TDQUB	TD queue update block	TDP
TDST	TD static storage	TDP
TDTE	Transaction dump table elements	DU
TDUA	TD UOW anchor block	TDP
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
TOGGLE	Feature toggle list	PA
TRA	Trace domain anchor block	TR
TRDCB	Auxiliary trace data set DCB	TR
TRDECB	Auxiliary trace data set DECB	TR

CICS control block		VERBEXIT keyword
TRE	Timer request elements	TI
TS	User temporary storage	AP APS
TSA	TS anchor block	TS
TSBUFFER	TS I/O buffer	TS
TSHANCH	TS shared class anchor	TS
TSI	TS item descriptor	TS
TSM	TS main item header	TS
TSMNANCH	TS main class anchor	TS
TSMODEL	TS model class anchor	TS
TSNANCH	TS name class anchor	TS
TSOANCH	TS ownership lock class anchor	TS
TSQ	TS queue control block	TS
TSQANCH	TS queue class anchor	TS
TSS	TS aux section descriptor	TS
TST	TS table header	TS
TSTTE	TS table entry	TS
TSW	TS wait element	TS
TSX	TS aux item descriptor	TS
TXD64	Transaction definition instance 64 bit extension	PCT
TXDINST	Transaction definition instance	PCT
TXDSTAT	Transaction definition static data	PCT
TXN	XM domain transaction	XM
UCA	Use count array	MRO
UET	User exit table	UEH
USER24	Task storage - below 16 MB, user key	AP APS
USER31	Task storage - above 16 MB, user key	AP APS
USER64	Task storage - above the bar, user key	AP APS
VCA	TS VSWA control area	TS
VEMA	TD VSAM error message area	TDP
VSWA	VSAM work area	FCP
VSWA	TS VSAM work area	TS
WAITLST	Wait list	TCP
WBABC	Web anchor block	WB
WBSTC	Web state manager block	WB
WCB	Web service control block	PI

CICS control block		VERBEXIT keyword
WHB	Web service header block	PI
WQP	Domain wait queue	DM
WRB	Web service resource block	PI
XMA	XM domain anchor block	XM
XM_XB	XM domain browse element	XM
XRH	TS aux record header	TS
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

Summary data for PG and US keywords

This reference lists the elements of the control block summaries in a level-1 IPCS 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 Chapter 8, "SDUMP contents and IPCS CICS VERBEXIT keywords," on page 263 form=numonly, 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.

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 - OS/VS COBOL (programs of this type do not run in CICS)
- CO2 - Enterprise COBOL or VS COBOL II
- LE3 - Le370
- PLI - PL/I.

LANG DED
Language deduced, one of the following:

- NDD - Not deduced
- ASS - Assembler
- C - C
- COB - OS/VS COBOL (programs of this type do not run in CICS)
- CO2 - Enterprise COBOL or VS COBOL II
- LE3 - Le370
- 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 16 MB).

EXEC KEY
Execution key, either C (CICS) or U (user).

DPL SUBS
DPL subset, either DP (DPL subset) or F (full API).

RE LOAD
Indicates 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.

CHCB Summary

CHANNEL
Channel name (followed by *CURRENT* if it is the program's current channel).

CHCB
CHCB address.

LEN
Total length of all containers in the channel.

CCSID
Default coded character set ID for the channel.

GN
Generation number.

CPCB
Address of container pool control block.

CRCB Summary

CONTAINER
Container name.

TYPE
Container type. The type is one of the following:

CICS
An internal system container.

R/O
A read-only container.

USER
A user-data container.

CRCB
CRCB address.

LEN
Length of data in the container.

CCSID
The default coded character set ID for the container or, if the container was created with the BIT option, DTYPE(BIT).

GN
Generation number.

CSCB
CSCB anchor 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 module.

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) as defined in the CICS RACF segment. This specifies when an inactive terminal is signed off.

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 United States of America. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 rights 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information 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.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

Licensees of this program who want 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

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 Client Relationship Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

IBM CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 (CICS TS 5.4) are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- Securing overview
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS TS 5.4, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS TS 5.4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services

- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- CICS Transactions
- CICSPlex System Manager (CICSPlex SM) Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS TS 5.4, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information (PII) is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect PII. If this Software Offering uses cookies to collect PII, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer®:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user's user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect PII from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z