Your task is waiting on a physical terminal

Investigating storage waits

Was the request for too much storage?

Is the storage close to being exhausted?

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

Investigating temporary storage waits

Resource type ENQUEUE

Resource type TSAUX

Resource type TSBUFFER

Resource type TSEXTEND

Resource type TSIO

Resource type TSPOOL

Resource type TSQUEUE

Resource type TSSHARED

Resource type TSSTRING

Resource type TSWBUFFR

Investigating enqueue waits

Using a system dump to resolve enqueue waits

EXEC CICS ENQ waits

Investigating interval control waits

Finding the reason for a DELAY request not completing

Using trace to find out why tasks are waiting on interval control

Investigating file control waits

Resource type CFDTWAIT—wait for CFDT request to complete

Resource type CFDTPOOL - wait for CFDT a request slot

Resource type CFDTLRSW - wait for CFDT locking request slot

Resource type FCACWAIT & FCCRWAIT—wait for SMSVSAM cleanup

Resource type FCBFWAIT—waits for VSAM buffers

Resource type FCCAWAIT—waits on the SMSVSAM control ACB

Resource type FCCFOR—wait for SMSVSAM server notification

Resource type FCCFQS—wait for user task to issue quiesce

Resource type FCDWWAIT—wait for VSAM to complete update processing

Resource type FCFSWAIT—wait for file state changes

Resource type FCIOWAIT—wait for VSAM I/O (non-RLS)

Resource type FCIRWAIT—wait for FC environment to be rebuilt

Resource types FCPSWAIT and FCSRSUSP—waits for VSAM strings

Resource type FCQUIES—wait for a quiesce request to complete

Resource type FCRAWAIT—file control to process non-recoverable requests

Resource type FCRBWAIT—file control to process recoverable requests

Resource type FCRDWAIT—wait for a drain of the RLS control ACB

Resource type FCRPWAIT—wait for file control initialization to complete

Resource Type FCRRWAIT—wait for a drain of the RLS restart to complete

Resource type FCRVWAIT—wait for VSAM I/O (RLS)

Resource type FCITISUSP—wait for a VSAM transaction ID

Resource type FCXSWAIT—VSAM exclusive control wait

Resource type ENQUEUE—waits for locks on files or data tables

Investigating loader waits

Investigating lock manager waits

Collecting the evidence

Investigating transaction manager waits

Maximum task condition waits

Transaction summary

MXT summary

Transaction class summary

A user task is waiting on resource type FOREVER

Resource type TRANDEF
### Chapter 7. How tasks are made to wait

The resources on which tasks in a CICS system can wait.

**Dispatcher waits**
- Resource name JVM_POOL
- Resource name OPENPOOL
- Resource name OPEN_DEL
- Resource type DSTSKDEF
- Resource type SOSMVS

**CICS DB2 waits**
- Resource type CDB2CONN
- Resource type CDB2RDYQ
- Resource type CDB2TCB
- Resource type DB2
- Resource type DB2_INIT
- Resource type DB2EDISA

**DBCTL waits**
- Connection to DBCTL has failed to complete
- A user task is waiting on resource type DBCTL
- Disconnection from DBCTL has failed to complete

**EDF waits**

**Log manager waits**
- Resource type LG_DEFER
- Resource type LG_FORCE
- Resource type LGDELALL
- Resource type LGDELTRAN
- Resource type LGEDBLK
- Resource type LGENDCRS
- Resource type LGREDBLK
- Resource type LGREDCRS
- Resource type LGSTRBLK
- Resource type LGSTRCRS
- Resource type LGWRITE

**Task control waits**
- Resource type KCCOMPAT
- Resource type KC_ENQ

**VTAM terminal control waits**
- Resource type ZC
- Resource type ZC_ZCGRP
- Resource type ZC_ZGCH
- Resource type ZC_ZGIN
- Resource type ZC_ZGPR, resource name PSINQECB
- Resource type ZC_ZGPR, resource name PSOP2ECB
- Resource type ZC_ZGUB
- Resource type ZCIOWAIT
- Resource type ZCZGET
- Resource type ZCZNAC
Resource type ZXQOWAIT .............................. 134
Resource type ZXSTWAIT .............................. 134
Interregion and intersystem communication waits ............ 134
IIOP waits ...................................... 135
Transient data waits ................................ 135
    Resource type TD_INIT—waits during initialization processing 135
    Resource type TDEPLOCK—waits for transient data extrapartition requests 136
    Resource types TDIPLOCK, ENQUEUE, TD_READ, Any_MBCB,
        Any_MRCB, MBCB_xxx, and MRCB_xxx 136
XRF alternate system waits ...................... 139
    XRSTIECB ................................ 139
    XRSTIAECB ............................... 139
    XRSTCECB ................................ 139
    XRSSRAECB .............................. 140
    XRSSSECBI ............................... 140
    XRSSSTECB ............................... 140
CICS system task waits ..................... 140
    FEPI waits ................................ 141
Recovery manager waits ...................... 141
    Resource type RMCLIENT .................. 142
    Resource type RMUOWOBJ, resource name LOGMOVE 142
    Resource type RMUOWOBJ, resource name EXISTENC 142
CICS Web waits ............................. 142
    Resource type WBALIAS .................. 142

Chapter 8. Dealing with loops .................. 143
What sort of loop is indicated by the symptoms? .......... 143
    Tight loops and non-yielding loops .......... 144
    Yielding loops .......................... 145
Investigating loops that cause transactions to abend with abend code AICA 146
    The documentation you need .............. 146
    Looking at the evidence .................. 147
    Identifying the loop ..................... 147
    Finding the reason for the loop .......... 147
Investigating loops that are not detected by CICS .... 157
    The documentation you need .............. 158
    Identifying the loop ..................... 158
    Finding the reason for the loop .......... 158
What to do if you cannot find the reason for a loop .... 158
    Investigating loops using interactive tools 159
    Modifying your program to investigate the loop 159

Chapter 9. Dealing with performance problems .... 161
Finding the bottleneck .......................... 161
    Initial attach to the transaction manager 161
    Initial attach to the dispatcher .......... 162
Why tasks fail to get attached to the transaction manager .. 162
Why tasks fail to get attached to the dispatcher ........ 163
    Is the MXT limit preventing tasks from getting attached? 163
Why tasks fail to get an initial dispatch .............. 165
    Priorities of tasks ...................... 165
Why tasks take a long time to complete .............. 166
    The effect of system loading on performance 167
    The effect of task time-out interval on performance 167
    The distribution of data sets on DASD volumes 167
A summary of performance bottlenecks, symptoms, and causes 167
Chapter 10. Dealing with incorrect output

Trace output is incorrect ........................................ 169
Tracing has gone to the wrong destination ................. 169
You have captured the wrong trace data ................... 170
Dump output is incorrect ......................................... 173
The dump does not seem to relate to your CICS region .... 173
You do not get a dump when an abend occurs .......... 173
Some dump IDs are missing from the sequence of dumps . 175
You do not get the correct data when formatting the CICS system dump 176
Incorrect data is displayed on a terminal .................... 176
The preliminary information you need to get .............. 177
Tools for debugging terminal output in a VTAM environment ...... 177
Specific types of incorrect output for terminals .......... 178
Incorrect data is present on a VSAM data set ............. 182
An application does not work as expected ............... 183
General points for you to consider ....................... 183
Your transaction produces no output at all ............... 184
  Are there any messages explaining why there is no output? . 184
  Can you use the terminal where the transaction should have started? ... 184
  No output—what to do if the task is still in the system. 185
  No output—what to do if the task is not in the system. 185
  Did the task run? Techniques for all tasks ........ 185
  Investigating tasks initiated by ATI .................. 188
Your transaction produces some output, but it is wrong .... 189
  The origins of corrupted data ......................... 189
  Are records in the file incorrect or missing? .......... 190
  Is the data mapped correctly into the program? ....... 190
  Is the data being corrupted by bad programming logic? ... 190
  Is the data being mapped incorrectly to the terminal? ... 191

Chapter 11. Dealing with storage violations ................. 193
Avoiding storage violations .................................... 193
Two kinds of storage violation ................................ 194
CICS has detected a storage violation .................... 194
  What happens when CICS detects a storage violation .... 195
  What to do if you cannot find what is overlaying the SAA . 197
Storage violations that affect innocent transactions .... 199
  A strategy for storage violations affecting innocent transactions ...... 199
  Procedure for resolving storage violations affecting innocent transactions 200
  What to do if you still cannot find the cause of the overlay ........ 200
  Programming errors that can cause storage violations .... 200
Storage recovery .......................................................... 201

Chapter 12. Dealing with external CICS interface (EXCI) problems .... 203

Chapter 13. Dealing with MRO problems .................... 205

Chapter 14. Dealing with log manager problems .......... 207
Categories of problem ............................................. 207
Exceeding the capacity of a log stream ............... 208
  System log .................................................. 208
  General logs .............................................. 208
How CICS checks for the availability of the MVS logger .... 208
Some conditions that cause CICS log manager error messages ... 209
  Message DFHLG0772. ....................................... 210
  Message DFHLG0002. ....................................... 212
## Part 3. Using traces and dumps in problem determination

<table>
<thead>
<tr>
<th>Chapter 15. Using traces in problem determination</th>
<th>225</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS tracing</td>
<td>226</td>
</tr>
<tr>
<td>Trace points</td>
<td>226</td>
</tr>
<tr>
<td>Trace levels</td>
<td>226</td>
</tr>
<tr>
<td>CICS exception tracing</td>
<td>228</td>
</tr>
<tr>
<td>User exception trace entries</td>
<td>228</td>
</tr>
<tr>
<td>CICS XRF tracing</td>
<td>229</td>
</tr>
<tr>
<td>Entry types</td>
<td>230</td>
</tr>
<tr>
<td>Program check and abend tracing</td>
<td>231</td>
</tr>
<tr>
<td>CICS VTAM exit tracing</td>
<td>231</td>
</tr>
<tr>
<td>Controlling CICS VTAM exit tracing</td>
<td>232</td>
</tr>
<tr>
<td>Interpreting CICS VTAM exit trace entries</td>
<td>232</td>
</tr>
<tr>
<td>VTAM buffer tracing</td>
<td>232</td>
</tr>
<tr>
<td>Selecting tracing by transaction</td>
<td>233</td>
</tr>
<tr>
<td>Tracing for selected tasks</td>
<td>233</td>
</tr>
<tr>
<td>The tracing logic used by CICS</td>
<td>235</td>
</tr>
<tr>
<td>Selecting tracing by component</td>
<td>235</td>
</tr>
<tr>
<td>Component names and abbreviations</td>
<td>236</td>
</tr>
<tr>
<td>Defining tracing for JVMs</td>
<td>238</td>
</tr>
<tr>
<td>Selecting JVM trace options</td>
<td>239</td>
</tr>
<tr>
<td>Interpreting JVM trace</td>
<td>243</td>
</tr>
<tr>
<td>Selecting trace destinations and related options</td>
<td>244</td>
</tr>
<tr>
<td>CICS internal trace</td>
<td>245</td>
</tr>
<tr>
<td>CICS auxiliary trace</td>
<td>245</td>
</tr>
<tr>
<td>CICS GTF trace</td>
<td>246</td>
</tr>
<tr>
<td>Setting the tracing status</td>
<td>247</td>
</tr>
<tr>
<td>Setting the tracing status at system initialization</td>
<td>247</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. Formatting and interpreting trace entries</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreting extended-format CICS system trace entries</td>
<td>252</td>
</tr>
<tr>
<td>Interpreting short-format CICS system trace entries</td>
<td>255</td>
</tr>
<tr>
<td>Interpreting abbreviated-format CICS system trace entries</td>
<td>256</td>
</tr>
<tr>
<td>Interpreting user trace entries</td>
<td>258</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 17. Using dumps in problem determination</th>
<th>261</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting up the dumping environment</td>
<td>261</td>
</tr>
<tr>
<td>Detecting and avoiding duplicate system dumps</td>
<td>262</td>
</tr>
<tr>
<td>Where dumps are written</td>
<td>262</td>
</tr>
<tr>
<td>Events that can cause dumps to be taken</td>
<td>263</td>
</tr>
<tr>
<td>The ways that you can request dumps</td>
<td>263</td>
</tr>
<tr>
<td>The occasions when CICS requests a dump</td>
<td>264</td>
</tr>
</tbody>
</table>
CICS dumping in a sysplex ........................................ 265
Automatic dump data capture from related CICS regions ........ 265
Operator-requested simultaneous dump data capture .............. 266
Requesting dumps to resolve SMSVSAM problems ................ 267
Useful CICS master terminal and MVS console commands in a sysplex 267
Enabling system dumps for some CICS messages ................. 273
System dump actions with messages DFHAP0001 and DFHSR0001 ... 274
The dump code options you can specify .......................... 275
Specifying the areas you want written to a transaction dump .... 276
Dump table statistics .............................................. 276
What happens to a dump request if there is no dump table entry? 277
The transaction dump table ...................................... 278
The system dump table ........................................... 280
Dumping a CFDT list structure ................................... 281
Dumping a named counter list structure .......................... 281
Dumping a shared temporary storage list structure .............. 282
The CSFE ZCQTRACE facility .................................... 283

Chapter 18. Formatting and interpreting dumps .................... 285
Formatting transaction dumps .................................... 285
Formatting system dumps ........................................ 285
The DFHIPCSP CICS exit control data ............................ 285
A summary of system dump formatting keywords and levels .... 287
The default SDUMP formatting levels ............................. 296
Interpreting transaction dumps ................................ 296
Job log ......................................................... 297
Symptom string ................................................. 297
CICS Transaction Server for z/OS level ......................... 297
Transaction environment summary .............................. 297
Remote abend indicator ........................................ 297
PSW ......................................................... 297
Registers at the time of interrupt ............................... 297
Execution key ............................................... 297
Space ....................................................... 298
Registers at last EXEC command ................................ 298
Task control area (TCA) ....................................... 298
Task control area, system area .................................. 298
Transaction work area (TWA) .................................. 298
EXEC interface structure (EIS) ................................ 298
System EXEC interface block (SYSEIB) ......................... 298
EXEC interface user structure (EIUS) .......................... 298
EXEC interface block (DFHEIB) ............................... 298
Kernel stack entries .......................................... 298
Common system area (CSA) ..................................... 298
Common system area optional features list (CSAOPFL) ......... 299
Trace table (abbreviated format) ................................ 299
Trace table (extended format) .................................. 299
Common work area (CWA) ..................................... 299
Transaction storage ............................................. 299
Terminal control table terminal entry (TCTTE) ................. 300
Program information for the current transaction .............. 300
Module index ............................................... 300
Locating the last command or statement ...................... 300
Last command identification ................................... 301
Last statement identification .................................. 301
Locating program data ........................................ 302
Dumps for C/370 programs ........................................... 302
Location of COBOL dumped areas .......................... 302
Storage freeze ......................................................... 303
Formatting a coupling facility data table pool dump ........... 303
Formatting a named counter pool dump ....................... 304
Formatting a shared temporary storage pool dump ........... 304

Chapter 19. The global trap/trace exit ...................... 305
Establishing the exit ............................................... 305
Information passed to the exit .................................. 305
Actions the exit can take ......................................... 306
Program check handling .......................................... 307
Coding the exit ...................................................... 307

Part 4. Working with IBM to solve your problem .......... 309

Chapter 20. IBM program support .............................. 311
When to contact the Support Center ......................... 311
Dealing with the Support Center ............................... 311
What the Support Center needs to know ...................... 312
What happens next ................................................ 313
Reporting a FEPI problem to IBM ............................. 314

Chapter 21. APARs, fixes, and PTFs .......................... 315
The APAR process .................................................. 315
Collecting the documentation for the APAR ................. 315
General documentation needed for all problems with CICS 316
Sending the documentation to the change team .............. 316
Packing and mailing the APAR box ............................. 316
Applying the fix .................................................... 317
The APAR becomes a PTF ........................................ 317

Part 5. Appendixes ................................................ 319

Appendix A. SDUMP contents and IPCS CICS VEREXIT keywords .... 321
Finding the control blocks from the keywords .............. 321
Finding the keywords from the control blocks .......... 330

Appendix B. Summary data for PG and US keywords .... 339
PG keyword .......................................................... 339
PGA (program manager anchor) ............................... 339
System LLE Summary ............................................. 339
PGWE Summary .................................................... 340
PPTE Summary ..................................................... 340
PTA Summary ....................................................... 341
Task LLE Summary ................................................. 341
CHCB Summary ..................................................... 341
CRCB Summary ..................................................... 341
Task PLCB Summary ............................................... 341
US keyword ........................................................ 342
USXD summary .................................................... 342
USUD summary .................................................... 342

Bibliography ......................................................... 343
The CICS Transaction Server for z/OS library ............ 343

Contents xi
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The entitlement set</td>
<td>343</td>
</tr>
<tr>
<td>PDF-only books</td>
<td>343</td>
</tr>
<tr>
<td>Other CICS books</td>
<td>345</td>
</tr>
<tr>
<td>Determining if a publication is current</td>
<td>345</td>
</tr>
<tr>
<td>Accessibility</td>
<td>347</td>
</tr>
<tr>
<td>Index</td>
<td>349</td>
</tr>
<tr>
<td>Notices</td>
<td>369</td>
</tr>
<tr>
<td>Trademarks</td>
<td>370</td>
</tr>
<tr>
<td>Sending your comments to IBM</td>
<td>371</td>
</tr>
</tbody>
</table>
Preface

What this book is about

This book is about methods of determining the causes of problems in a system that uses CICS®. It contains information about resolving CICS application and system problems, dealing with the IBM® Support Center, and handling authorized program analysis reports (APARs).

- This book does not describe methods of problem determination for Java™ applications and enterprise Java beans in CICS. See Java Applications in CICS.

Note: For problem determination of the ONC/RPC feature, see the CICS External Interfaces Guide.

Who this book is for

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

What you need to know to understand this book

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

How to use this book

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

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

Table 1. Road map for the CICS Problem Determination Guide

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go through some preliminary checks</td>
<td>Chapter 1, “Introduction to problem determination,” on page 3</td>
</tr>
<tr>
<td>Classify the problem according to its symptoms</td>
<td>Chapter 2, “Classifying the problem,” on page 7</td>
</tr>
<tr>
<td>Look for information to help you diagnose and resolve the problem</td>
<td>Chapter 3, “Sources of information,” on page 17</td>
</tr>
<tr>
<td>Resolve transaction abnormal terminations.</td>
<td>Chapter 4, “Dealing with transaction abend codes,” on page 25</td>
</tr>
<tr>
<td>Resolve system abnormal terminations.</td>
<td>Chapter 5, “Dealing with CICS system abends,” on page 37</td>
</tr>
<tr>
<td>Decide whether the problem is caused by a wait, a loop, or a performance problem.</td>
<td>“Distinguishing between waits, loops, and poor performance” on page 12</td>
</tr>
<tr>
<td>Resolve problems caused by waits.</td>
<td>Chapter 6, “Dealing with waits,” on page 49</td>
</tr>
<tr>
<td>Resolve problems caused by loops.</td>
<td>Chapter 8, “Dealing with loops,” on page 143</td>
</tr>
</tbody>
</table>
Table 1. Road map for the CICS Problem Determination Guide (continued)

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolve problems caused by performance problems.</td>
<td>Chapter 9, &quot;Dealing with performance problems,&quot; on page 161</td>
</tr>
<tr>
<td>Know what to do if you don’t get the output you expected.</td>
<td>Chapter 10, &quot;Dealing with incorrect output,&quot; on page 169</td>
</tr>
<tr>
<td>Resolve problems caused by storage violations.</td>
<td>Chapter 11, &quot;Dealing with storage violations,&quot; on page 193</td>
</tr>
<tr>
<td>Resolve problems involving XRF.</td>
<td>CICS/ESA 4.1 Problem Determination Guide</td>
</tr>
<tr>
<td>Resolve problems with the external CICS interface.</td>
<td>Chapter 12, &quot;Dealing with external CICS interface (EXCI) problems,&quot; on page 203</td>
</tr>
<tr>
<td>Resolve problems with MRO.</td>
<td>Chapter 13, &quot;Dealing with MRO problems,&quot; on page 205</td>
</tr>
<tr>
<td>Resolve problems with log manager.</td>
<td>Chapter 14, &quot;Dealing with log manager problems,&quot; on page 207</td>
</tr>
<tr>
<td>Use CICS trace.</td>
<td>Chapter 15, &quot;Using traces in problem determination,&quot; on page 225</td>
</tr>
<tr>
<td>Use CICS dump.</td>
<td>Chapter 17, &quot;Using dumps in problem determination,&quot; on page 261</td>
</tr>
<tr>
<td>Report the problem to IBM.</td>
<td>Part 4, &quot;Working with IBM to solve your problem,&quot; on page 309</td>
</tr>
<tr>
<td>Look up a dump exit keyword.</td>
<td>Appendix A, &quot;SDUMP contents and IPCS CICS VERBEXIT keywords,&quot; on page 321</td>
</tr>
</tbody>
</table>

Notes about terms used in this book

When the term “CICS” is used without any qualification in this book, it refers to the CICS element of CICS Transaction Server for z/OS®, Version 3 Release 1.

“MVS” is used for the operating system, which is an element of z/OS.

Throughout this book, the term APPC is used to mean LUTYPE6.2. For example, APPC session is used instead of LUTYPE6.2 session.
Summary of changes


Changes for CICS Transaction Server for z/OS, Version 3 Release 1

Technical changes
• Because of the removal of run-time support for Java program objects and hot-pooling (HPJ), and of the open TCB mode H8, which was used for hot-pooling Java program objects, the resource type HP_POOL is no longer a cause of dispatcher waits.

Structural changes
• There are no significant structural changes.

Changes for CICS Transaction Server for z/OS, Version 2 Release 3

Technical changes
• A new section on JVM tracing has been added. See “Defining tracing for JVMs” on page 238.

Structural changes
• There are changes to the way that information is organised in each chapter.

Changes for CICS Transaction Server for z/OS, Version 2 Release 2

Technical changes
• “Dumping a shared temporary storage list structure” on page 282 has been added.
• New information has been added to “The resources on which tasks in a CICS system can wait” on page 111.
• Information which is specific to OS/390 Release 3 and earlier releases has been removed from Chapter 14, “Dealing with log manager problems,” on page 207.
• “How CICS checks for the availability of the MVS logger” on page 208 has been expanded.
• Reference information associated with dump and trace has been added for the following components and domains:
  – 3270 Bridge (BR)
  – Enterprise Java domain (EJ)
  – ECI over TCP/IP domain (IE)
  – IIOP domain (II)
  – JVM domain (SJ)
  – Object transaction services domain (OT)
  – Partner domain (PT)
  – Request streams domain (RZ).
• “Some conditions that cause CICS log manager error messages” on page 209 has been added to Chapter 14, “Dealing with log manager problems,” on page 207.
Changes for CICS Transaction Server for z/OS, Version 2 Release 1

A number of small amendments have been made to this edition, the most significant being the changes in the trace entries.

For problem determination guidance associated with CICS business transactions, see CICS Business Transaction Services.

Changes for CICS Transaction Server for OS/390, Version 1 Release 2

The main changes for this edition are as follows:

- "Dealing with a corrupt system log" on page 221 describes the new diagnostic run mechanism for diagnosing problems with the CICS system log.

Changes for CICS Transaction Server for OS/390, Version 1 Release 1

The main changes for this edition are as follows:

- New section dealing with problem determination in the log manager area.
- New information in the ‘Dealing with waits’ section detailing waits in the following areas:
  - Log manager
  - EDF
  - Enqueue facility
  - Recovery manager.
- Introduction of the WLM_WAIT_TYPE parameter on DSSR functions.
- Purge status information included for suspended tasks.
- The trace entry interpretation tables have been moved to the Diagnosis Reference.

Changes for CICS for MVS/ESA 4.1

The main changes for this edition are as follows:

- Information added regarding transaction isolation and the data integrity protection that it provides.
- New information in the ‘Dealing with waits’ section to reflect transaction isolation, and other new wait types.
- Discussion of simultaneous dump capture in multiple CICS systems across a sysplex.
- Reference information added to include the new dumped and traced data from the following areas:
  - Transaction isolation
  - VTAM persistent session support
  - XCF/MRO
  - External CICS interface
  - Directory domain
  - Program manager domain
  - Security manager domain
  - Transaction manager domain
  - User domain.
Part 1. Approaches to problem determination

This part contains:

- Chapter 1, “Introduction to problem determination,” on page 3
- Chapter 2, “Classifying the problem,” on page 7
- Chapter 3, “Sources of information,” on page 17
Chapter 1. Introduction to problem determination

This book tells you how to find the reasons for problems with your CICS system.

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

Sometimes, you cannot solve the problem yourself if, for example, it is caused by limitations in the hardware or software you are using. If the cause of the problem is CICS code, you need to contact IBM, as described in Part 4, “Working with IBM to solve your problem,” on page 309.

This section contains the following topics:

• “Before you start—preliminary checks”
• “What to do next” on page 6

Before you start—preliminary checks

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

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

1. Has the CICS system run successfully before?
   If the CICS system has not run successfully before, it is possible that you have not yet set it up correctly. You can check that CICS installed correctly by running batch or online verification procedures. See CICS Transaction Server for z/OS Installation Guide for more information. If you have verified that CICS installed successfully, check the appropriate migration guide for any possible impacts to your system. The migration guides available are.....
   If you are currently migrating to CICS Transaction Server for z/OS, Version 3 Release 1, ensure that you are aware of all the changes that have been made for this release. For details of these, see the CICS Transaction Server for z/OS Migration from CICS TS Version 2.3

2. Are there any messages explaining the failure?
   If a transaction abends, and the task terminates abnormally, CICS sends a message reporting the fact to the CSMT log (or your site replacement). If you find a message there, it might immediately suggest a reason for the failure.
   Were there any unusual messages associated with CICS start up, or while the system was running before the error occurred? These might indicate some system problem that prevented your transaction from running successfully.
   If you see any messages that you do not understand, you can 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 and Codes 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 have defined sufficient resource for the application, such as VSAM file strings. Typically, if you had not defined sufficient resources, you would find that the problem is related to the number of users of the application.

b. Are you using exit programming interface (XPI) calls? If so, be sure to observe the XPI protocols and restrictions exactly. For programming information about the XPI, see the CICS Customization Guide. The exit programming interface enables you to invoke a domain and enter its environment directly; using it incorrectly can cause severe CICS system problems. Here are some particular points for your attention:
   • Are the input parameters correct? If their format is not valid, they are rejected by the called domain, and an exception trace is made. If their values are acceptable to the domain but inappropriate for the system, they could cause unpredictable effects.
   • Be aware that you cannot use some XPI calls within some of the user exits. If you do, the results can be unpredictable, and can cause CICS to stall or abend. The CICS Customization Guide tells you which exits can use X calls and which cannot.

c. Consider your CICS system definition parameters if the problem is not related to any particular application. Poorly defined parameters may be the cause of problems in your system. You can find guidance about setting up your CICS system in the CICS System Definition Guide.

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

4. Does the failure occur at specific times of day?
   If the failure occurs at specific times of day, it could be dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so those are the times when load-dependent failures are most likely to happen. 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 have been deleted from the system long before the symptoms are seen.

   A method you can use to investigate random overlays is described in Chapter 11, “Dealing with storage violations,” on page 193.

6. Have any changes been made since the last successful run?
   Service
   a. Have you applied a PTF to CICS?
   b. Did it install successfully or did you get an error message during installation? If you installed it successfully, check with IBM for any PTF error.
   c. Have any patches applied to any other program affected the way CICS interfaces with the program?

   Hardware
a. Have you changed your hardware?

Software
a. Have you changed your software?
b. If you have installed a new or modified application, check for error messages in the output from the following:
   • Translator
   • Compiler
   • Assembler
   • Linkage editor

Administration
a. Have you changed your initialization procedure, for example by JCL, CICS system initialization or override parameters, or VTAM® CONFIG/LIST?
b. Has CICS generated any error messages during initialization?
c. Have you installed any resource definitions defined using CEDA?
   If the definitions were made but not installed when CICS was last terminated, they might not have been preserved over the termination and subsequent start up. In general, changes made to the CSD but not installed are not visible when the CICS system is warm started. However, if the change was in a group in the GRPLIST specified on a cold start, it is effectively installed during startup. (Changes which have been installed are not visible after a cold start unless they were made to a group in the GRPLIST.)
   If START=AUTO was specified in the system initialization table, or as an override, you need to examine the job log to find out how the CICS system last came up.
   For detailed guidance about the ways in which resources can be defined and installed, see the CICS Resource Definition Guide.

7. Are specific parts of the network affected by the problem?
a. Can you identify specific parts of the network that the problem affects? If you can, look for any explanatory message from the access method. Even if no message has been sent to the console, you might find one in the CSNE log.
b. Have you made any network-related changes?
c. If the problem affects a single terminal, are your terminal definitions correct? Consider both the 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 has probably stalled. See "What to do if CICS has stalled" on page 106 for advice about dealing with CICS system stalls.

8. Has the application run successfully before?
a. Have any changes been made to the application since it last ran successfully?
   Examine the new or modified part of the application.
b. Have you used RDO to create or alter a transaction, program, or 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

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

d. Have all the functions of the application been fully exercised before?
Establish what the program was doing when the error occurred, and check the source code in that part of the program.
If a program has been run successfully on many previous occasions, examine the contents of any records, screen data, and files that were being processed when the error occurred. They may contain some unusual data value that 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.

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

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.

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

If you have not yet found the cause, you must start to look at the problem in greater detail. Begin by finding the best category for the problem, using the approach described in Chapter 2, “Classifying the problem,” on page 7.
Chapter 2. Classifying the problem

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

- “Using symptom keywords as a basis for classifying problems”
- “Using the symptoms to classify the problem” on page 8
- “Distinguishing between waits, loops, and poor performance” on page 12
- “Classifying problems by functional area” on page 16
- “What to do next” on page 16

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

Using symptom keywords as a basis for classifying 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.

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

For software problems, the classifications you will see are:

- **ABEND** (for transaction abends, see Chapter 4, “Dealing with transaction abend codes,” on page 25; for system abends, see Chapter 5, “Dealing with CICS system abends,” on page 37)
- **WAIT** (see Chapter 6, “Dealing with waits,” on page 49)
- **LOOP** (see Chapter 8, “Dealing with loops,” on page 143)
- **POOR PERFORMANCE**, or **PERFM** (see Chapter 9, “Dealing with performance problems,” on page 161)
- **INCORRECT OUTPUT**, or **INCORROUT** (see Chapter 10, “Dealing with incorrect output,” on page 169)
- **MESSAGE**

All but the last of these, MESSAGE, are considered in this book. If you receive a CICS error message, you can use the CICS message transaction, CMAC, for online message information. If you do not have access to a running CICS system, look in [CICS Messages and Codes](#) for an explanation. If you get a message from another IBM program, or from the operating system, you need to look in the messages and codes book from the appropriate library for an explanation of what that message means.

[CICS Messages and Codes](#) might give you enough information to solve the problem quickly, or it might redirect you to other information sources for further guidance. If you are unable to deal with the message, you may eventually need to contact the IBM Support Center for help.

One type of problem that might give rise to a number of symptoms, usually ill-defined, is that of poor application design. Checking the design of an application is beyond the scope of this book, but one instance is described in “Poor application design” on page 16.
design” on page 15, where you will find there an example of how bad design can give rise to an application with poor usability.

Using the symptoms to classify the problem

The paragraphs that follow are to help you to 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.

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 9 to confirm this before going on to Chapter 9, “Dealing with performance problems,” on page 161 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 and Codes](#) 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/bbb) has occurred at offset X'offset in module modname.

DFHSR0601 Program interrupt occurred with system task taskid in control
If you get either of these messages, or any others for which the system action is to terminate CICS, turn to Chapter 5, “Dealing with CICS system abends,” on page 37 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 Chapter 6, “Dealing with waits,” on page 49 and Chapter 8, “Dealing with loops,” on page 143, 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, see the CICS Performance Guide.

You can find guidance about identifying specific performance bottlenecks in your CICS system in Chapter 9, “Dealing with performance problems,” on page 161.
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 Chapter 9, “Dealing with performance problems,” on page 161 for further guidance.

A task is running slowly

If just one task is running slowly, it is likely that the explanation lies with the task itself. It could be in a loop, or it could periodically be entering a wait state. You need to decide which of these possibilities is the most likely before starting systematic problem determination. The ways that you might distinguish between waits and loops are described in “Distinguishing between waits, loops, and poor performance” on page 12.

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

A task stops running at a terminal

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

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

First, make sure that the task is still in the system. Use CEMT INQ TASK to check its status, and make sure that it has not simply 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:

- CDNC, 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 DFHZNCP

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

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

Read “Distinguishing between waits, loops, and poor performance” on page 12 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 and Codes for an explanation of the message, and, perhaps, advice about what you should do to solve the problem. If the code is not there, or the explanation or advice given is not sufficient for you to solve the problem, turn to Chapter 4, “Dealing with transaction abend codes,” on page 25.

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 and Codes 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 manual:

- **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 Chapter 10, “Dealing with incorrect output,” on page 169.

A storage violation has occurred

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

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

If you see this message, or you know (through other means) that a storage violation has occurred, turn to Chapter 11, “Dealing with storage violations,” on page 193 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.
An XRF error has occurred

If an XRF error has occurred, turn to the CICS/ESA 4.1 Problem Determination Guide for advice on what to do.

Distinguishing between waits, loops, and poor performance

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

Any of the following symptoms could be caused by a wait, 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 106.

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 Chapter 6, “Dealing with waits,” on page 49 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 Chapter 8, “Dealing with loops,” on page 143.

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 106.

Consider the evidence you have so far. Does it indicate a loop? If so, turn to Chapter 8, “Dealing with loops,” on page 143, where there are procedures for defining the limits of the loop.

**Poor performance**

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

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

If you get many messages telling you 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.
You see one of the following messages when CICS is under stress in one of the DSAs:

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

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

If there is no such indication, see Chapter 9, “Dealing with performance problems,” on page 161 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.

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

**Environment:**

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

**Symptoms:**

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

**Explanation:**

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

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

Note that these symptoms 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.
Classifying problems by functional area

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

- **XRF errors** (see the *CICS/ESA 4.1 Problem Determination Guide*)
- **EXCI problems** (see Chapter 12, “Dealing with external CICS interface (EXCI) problems,” on page 203)
- **MRO problems** (see Chapter 13, “Dealing with MRO problems,” on page 205)
- **Log manager problems** (see Chapter 14, “Dealing with log manager problems,” on page 207)
- **Java and EJB problems** (see *Java Applications in CICS*)
- **Storage violations** (see Chapter 11, “Dealing with storage violations,” on page 193)

Whereas XRF, EXCI, and MRO errors can easily be classified in a straightforward way, confirming that you have a storage violation can be difficult. Unless you get a CICS message stating explicitly that you have a storage violation, you could get almost any symptom, depending on what has been overlaid. You might, therefore, classify it initially as one of the RETAIN symptom types described in “Using symptom keywords as a basis for classifying problems” on page 7.

What to do next

If you have already decided that you should refer the problem to the IBM Support Center, you can find advice about dealing with the Center in Chapter 20, “IBM program support,” on page 311.
Chapter 3. Sources of information

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

- "Your own documentation"
- "Product information"
- "Source listings and link-edit maps"
- "Abend codes and error messages" on page 18
- "Symptom strings" on page 18
- "Change log" on page 18
- "Dumps" on page 18
- "Statistics" on page 19
- "Monitoring" on page 19
- "Transaction inputs and outputs" on page 19
- "Traces" on page 21

Your own 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. 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.

Product information

Product information can refer to the CICS Information Center, or libraries for any other products you use with your application.

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

Source listings and link-edit maps

Include the source listings of any applications written at your installation 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 microfiche or CD-ROM.) Make sure you include the relevant linkage-editor output with your source listings to avoid wasting time trying to find your way through 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 the [CICS System Definition Guide](#). 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 and Codes](#) and are available in the CICS Information Center. 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.

**Dumps**

Dumps are an important source of detailed information about problems. Whether they are the result of an abend or a user request, they allow you to see a snapshot of what was happening in CICS at the moment the dump is taken. Chapter 17, "Using dumps in problem determination," on page 261 contains guidance about using dumps to locate problems in your CICS system. However, because they do only provide a "snapshot", you may need to use them in conjunction with other sources of information relating to a longer period of time, such as logs, traces, and statistics.
Statistics

Statistics are often overlooked as a source of debugging information, but those that relate to an application program can help solve problems. It is useful to have a statistical profile (as mentioned in "Your own 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 may indicate the source of a problem.

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

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

Monitoring

You can use CICS monitoring to provide information for debugging applications. In addition to the system-defined event monitoring points (EMPs) that already exist within CICS code itself, you can define user event monitoring points in your own application programs by using the EXEC CICS MONITOR 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 on the MCT see the CICS Resource Definition Guide. See the CICS Application Programming Reference for programming information on syntax and options of the MONITOR POINT command.

When your monitoring data has been collected, you can read it into a database using, for example, the Service Level Reporter Version 2 (SLR II).

See the CICS Performance Guide for guidance about choosing performance tools. See the CICS Supplied Transactions for information about the transactions needed to invoke them.

Transaction inputs and outputs

Transaction inputs and outputs can be divided into the following areas:
- Terminal data
- Transient data and temporary storage
- Passed information
- Files and databases
Terminal data

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

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

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

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

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 PF 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 nondisplay fields used to pass data that may not be protected?

Transient data and temporary storage

If the program explicitly uses any transient data or temporary storage queues, inspect them to see if their content is what you expect. You can use the CICS-supplied transaction, CEBR, to inspect temporary storage queues in some detail. See CICS Supplied Transactions 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.

**Traces**

CICS provides a tracing facility that enables you to trace transactions through the CICS components as well as through your own programs. CICS auxiliary trace enables you to write trace records on a sequential device for later analysis.

For information about the tracing facilities provided by CICS, read Chapter 15, "Using traces in problem determination," on page 225.
Part 2. Dealing with the problem

This part contains:

- Chapter 4, “Dealing with transaction abend codes,” on page 25
- Chapter 5, “Dealing with CICS system abends,” on page 37
- Chapter 6, “Dealing with waits,” on page 49
- Chapter 8, “Dealing with loops,” on page 143
- Chapter 9, “Dealing with performance problems,” on page 161
- Chapter 10, “Dealing with incorrect output,” on page 169
- Chapter 11, “Dealing with storage violations,” on page 193
- Chapter 12, “Dealing with external CICS interface (EXCI) problems,” on page 203
- Chapter 13, “Dealing with MRO problems,” on page 205
- Chapter 14, “Dealing with log manager problems,” on page 207
Chapter 4. Dealing with transaction abend codes

This section gives guidance about finding the cause of transaction abends.

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 35).

**Collecting the evidence**

You should find all the evidence you need to investigate the transaction abend in the information sent to the various transient data queues for error messages, and in the transaction dump. If no transaction dump has been produced, it is possible that transaction dumping has been suppressed for the transaction (via the transaction definition), or the dump code entry in the transaction dump code table suppresses dumping. For guidance about changing the dumping options so that you get a transaction dump, see Chapter 17, “Using dumps in problem determination,” on page 261.

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

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

**Symptom string**

CICS produces a symptom string as part of the transaction dump. The symptom string gives some details about the circumstances of the transaction dump. It might show, for example, that the dump was taken because the transaction abended with the abend code ASRA. If you refer the problem that caused the dump to be taken to the IBM Support Center, they will use the symptom string to search the RETAIN database for problems similar to it. For an introduction to symptom strings and their contents, see "Looking at the symptom string in the dump" on page 38."
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 and Codes](#).

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 — AICA, ASRA, ASRB, ASRD and AEYD

The abend codes AICA, ASRA, ASRB, ASRD and AEYD are dealt with separately because special procedures apply to them. If your abend code is something other than these, use the procedures in [“Last statement identification” on page 301](#) to find the last command that was executed, and then turn to [“Analyzing the problem further” on page 35](#). The best source of information on CICS abends can be found in [CICS Messages and Codes](#). It contains a section that lists all transaction abend codes issued by CICS. There is an explanation of why the code was issued, followed by details of system and user actions. The same information is available online, using the CICS-supplied messages and codes transaction, CMAC.

If, after reviewing the material in [CICS Messages and Codes](#), you cannot find the cause of the problem, continue with the procedures outlined in [Chapter 4, “Dealing with transaction abend codes,” on page 25](#).

AICA abends

If your transaction terminated with abend code AICA, the transaction is likely to have been in a loop. You can find detailed guidance about dealing with loops in [Chapter 8, “Dealing with loops,” on page 143](#).

ASRA abends

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

A transaction can abend with an abend code of ASRB 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 outlined in "Locating the last command or statement" on page 300 to find the origin of the abend in your program. That information, together with the description and procedures for ASRB abends given in CICS Messages and Codes, should be sufficient for you to solve the problem.

ASRD abends

A transaction abends with code ASRD if:

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

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

AEYD abends

If command protection is activated by the CMDPROT(YES) 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 containing the invalid address. The trace should include an exception trace entry created by DFHEISR. This entry should identify the parameter in error. If the abend is handled, EXEC CICS ASSIGN ASRASTG, ASRAKEY, ASRASPC, and ASRAREGS can give additional information.

To prevent a recurrence of the abend, it is recommended that you correct the program code. Alternatively, changing one or more of the following options may alleviate the problem:
- EXECKEY in the 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 193.

Finding where a program check occurred

When a transaction abends with code ASRA or ASRD, the first thing you need to do is find out where the program check occurred. CICS will have attempted to establish this for you. A record of the program in error and the offset of the program check within the program load module are contained in the following places:
- Message DFHAP0001 or DFHSR0001, which 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 296.

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

If the offset appears as X'FFFFFFFF', CICS was unable to establish the location of the program check. If this is the case, use the PSW to obtain the next sequential instruction address. The PSW may be found in the following places:
• The TACB for the abend
• At the head of the formatted transaction dump
• Within the kernel error data block traced by exception trace point IDs AP 0781 or AP 0783

Now note down the start and end addresses of the different program areas in the transaction dump. Is the next sequential instruction address from the PSW in any of the programs? If so, then that is the program in which the interrupt occurred. Use the procedure described in “Locating the last command or statement” on page 300 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.

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 35.

What type of program check occurred?

Knowing what type of program check occurred can be helpful in finding the cause of the error. This is indicated by the program interrupt code (PIC), which you can find in the PSW at the start of the transaction dump. You can find information about the PSW in ESA/370 from the IBM Enterprise Systems Architecture/370 Principles of Operation.
PIC PIC explanation

1 Operation exception—incorrect operation attempted.

Some possible causes
- 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.

Some possible causes
- Incorrect branch to this code; may be due to:
  - Overlaid register save area
  - Program overlaid by data that contains the privileged operation code

3 Execution exception—you are not allowed to EXECUTE an EXECUTE instruction.

Some possible causes
- Incorrect branch to this code
- Incorrect register contents; may be due to:
  - Overlaid register save area
  - Program overlaid by data that contains the incorrect instruction
  - Incorrect program logic

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

Some possible causes
- Resource unavailable, and return code not checked. Program logic assumed valid address returned and took inappropriate action.
- Incorrect interface parameters to some other program or subsystem (for example, VSAM or DL/I).
- Overlaid register save area, causing incorrect reference to data.
- In an assembler-language program, incorrect initialization or modification of a register used to address data.
- Attempt to access internal control blocks illegally or 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, the ECDSA or the ERDSA.
- 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 executing in USER key may suffer a protection exception when attempting to access the USER key task-lifetime storage of another transaction.
• Storage, passed to CICS as an output parameter through the EXEC interface, that is not addressable by the application issuing the call. The transaction is abended AEYD, and the PSW shows that a protection exception has occurred.

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

Some possible causes
• Incorrect register contents; may be due to:
  – Overlaid register save area.

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

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

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

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

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

Some possible causes
• Incorrect user data
• Overlaid data areas
• Overlaid register save area, causing incorrect reference to data

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

Dealing with arithmetic exceptions

If the program check was due to an arithmetic error (interruption codes 7 through F), you need to find the operands used in the last instruction. Use the procedure described in section “Locating program data” on page 302 to locate the fields. You need to know a little about the type of arithmetic being done, so that you can tell if the operands are valid. The interrupt you received tells you what sort of arithmetic the system was doing (binary, packed decimal, or floating point), but you need to determine if that is what you had intended to do. You might need to consult a programming language manual if you have any queries about this.
When you have identified the operands, you need to decide where the problem is. Questions to consider include:

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

**Dealing with protection exceptions**

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

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

If 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 USDA 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 USDA or EUDSA, but the second transaction is defined with ISOLATE(YES). (For a full description of the transaction isolation facility and its use, see the [CICS Resource Definition Guide](#).

If any of these events occurs, CICS abnormally terminates the transaction with abend code ASRA and issues message DFHSR0622 which identifies the DSA over which the program attempted to write. This information is in the TACB and is traced by exception trace point ID AP 0781. It is also useful to know the execution key of the program at the time of the protection exception and whether the program was executing in a subspace (CDSA, ECDSA, RDSA, ERDSA, USDA, or EUDSA). This appears in the TACB, exception trace point ID AP 0781 and at the head of the formatted transaction dump.

If the command protection facility is enabled, a protection exception can occur if storage, passed to CICS as an output parameter through the EXEC interface, is not accessible for READ/WRITE by the program issuing the command. The program is passing to CICS storage that it cannot itself update, but it requires CICS to 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.
Note

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

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

Causes of protection exceptions

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

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

Programs might also be incorrectly link-edited as reentrant, 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 may 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 IBM CICS Interdependency Analyzer for z/OS 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 the Resource Definition Guide. For more information about the CICS Interdependency Analyzer, see the CICS Interdependency Analyzer for z/OS User's Guide and Reference.
When migrating from a CICS 3.3 system to CICS Transaction Server for z/OS, Version 3 Release 1, it may be helpful to alter all transaction definitions to include the ISOLATE(NO) option. Then gradually change the definitions to use ISOLATE(YES) as interdependencies are removed. This facilitates manageable migration to a system that takes full advantage of the transaction isolation facility.

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 migrating to a system with command protection enabled, EXEC commands that pass unauthorized storage are identified and can be corrected.

Possible causes of protection exceptions referencing CICS DSAs

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

- Issuing an MVS macro request. Most MVS macros and services are not supported in EXECKEY(USER) application programs. Use of unsupported macros and services may cause a failure if these macros or services attempt to reference MVS storage outside the CICS DSAs.
- Referencing storage obtained by an MVS GETMAIN request or another MVS macro. MVS storage obtained by these methods resides outside the CICS DSAs, and is therefore protected from user key programs.
- Using PL/I statements, COBOL verbs or compiler options that are not permitted in CICS application programs (see the [CICS Application Programming Guide](#) for details of prohibited language statements and compiler options). For example, the use of CALL with the RES compiler option, or a verb such as INSPECT, may also cause 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 may have worked, or at least may not have caused the application to fail. However, the use of these statements and options can have other effects on the overall execution of the CICS system, and should be removed where possible.

- Modifying the CWA when CWAKEY=CICS is specified as a system initialization parameter. In a user key program, this is an invalid reference to storage allocated from the CDSA or ECDSA.
- Modifying the TCTUA when TCTUAKEY=CICS is specified as a system initialization parameter. In a user key program this is an invalid reference to storage allocated from the CDSA or ECDSA.
- Issuing 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 EXECHEKEY(CICS). These are all examples of programs that modify global work areas 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 may 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 may highlight a program error or an interdependency between two transactions. The IBM CICS Interdependency Analyzer for z/OS can help to identify potential transaction interdependencies. Examples of transaction interdependencies are:

- A transaction may use EXEC CICS GETMAIN to obtain 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 may 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).

**Analyzing the problem further**

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

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

**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 the *CICS IMS Database Control Guide*.

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
3. **Look up the abend code**
   If you need further advice, continue with 4 on page 36
4. Is this an AICA abend?
   If it is, read Chapter 8, “Dealing with loops,” on page 143. If not, continue with 5.

5. Is this an ASRA abend?
   If it is, go to step 7. If not, continue with 6.

6. Is this an ASRD abend?
   If it is, continue with 7. If not, go to 14.

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. If not, go to 14.

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 300.

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 302), and go to step 15. If not, continue with 13.

13. Was the PIC a protection exception?
    If it was, read “Dealing with protection exceptions” on page 31. Go to 15.

14. Find the last statement executed
    See “Locating the last command or statement” on page 300.

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 the CICS Front End Programming Interface User’s Guide.
Chapter 5. Dealing with CICS system abends

The purpose of this section is to give guidance about gathering essential information about CICS system abends. If you have not yet done so, use the CMAC transaction or look in CICS Messages and Codes for an explanation of any message you may have received, because it could offer a straightforward solution to your problem.

If the abend was clearly caused by a storage violation, turn directly to Chapter 11, “Dealing with storage violations,” on page 193. You know when CICS has detected a storage violation, because it issues this message:

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

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

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

This section tells you how to find out all of these things, and contains the following topics:
- “The documentation you need”
- “Interpreting the evidence” on page 38
- “Looking at the kernel domain storage areas” on page 40
- “Using the linkage stack to identify the failing module” on page 45

The documentation you need

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

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

If a system dump was not taken when the abend occurred, you need to find out why. Use the procedure described in “You do not get a dump when an abend occurs” on page 173, and follow the advice given there. When you are sure that dumping is enabled for the appropriate system dump code, you need to recreate 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 285 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 (search for the eye-catcher \*\*\*KE) contains summary information about the error. The internal trace table (eye-catcher \*\*\*TR) contains the exception trace entry (if any) that was made at the time the error was detected.

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

You can format and print the dump offline. Details of how to do this are given in the CICS Operations and Utilities Guide.

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

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

Interpreting the evidence

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

Looking at the messages

Any messages that accompany a CICS system abend can sometimes point directly to the cause of the failure. For every case, advice about how to react to a message is given in CICS Messages and Codes.

Looking at the exception trace entry

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

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

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

For details of trace entries, see Chapter 15, “Using traces in problem determination,” on page 225.

Looking at the symptom string in the dump

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

The symptom string:

• Is written to SYS1.LOGREC
- Is issued as part of message DFHME0116
- Appears at the beginning of a CICS system dump.

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 2. The keywords are used at the IBM Support Center to discover duplicate problems, or problems that have already been reported by other users and for which a solution is available.

If you have the IBM INFORMATION/ACCESS licensed program, 5665-266, you can search the RETAIN database yourself.

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

Table 2. Symptom string keywords

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

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

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

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

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

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

Finding which tasks are associated with the error

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

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

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

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

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

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

In Figure 1 on page 40, you can see that kernel task number 0008 is shown to be in error.

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. Most of the time, only one task is shown to be running. 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. It may, for example, be suspended because it is waiting for some resource, or it may be ready to run but waiting for a TCB to become available.
- “KTCB”, referring to CICS control blocks corresponding 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.

You are almost certain to find that the task shown to be in error has a status of “***Running***”, as in the example of Figure 1 on page 40. Such a task would have been running at the time the error was detected.

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

Two of the columns in the kernel task summary are particularly important in solving problems that require the use of traces. They are the TRAN_# and KE_NUM columns. The TRAN_# column for a task can contain:
- A number that matches the task number in the corresponding trace
- “TCP” for the CICS terminal control task
- Other character entries for CICS system tasks (for example, a component identifier like “AP” for a CICS system task in the AP domain).

When you are working with trace output, you can use the number from the TRAN_# column to identify entries associated with a user task up to the point at which that task passes control to CICS. To identify the CICS processing associated with the user task, you need to use the entry in the KE_NUM column of the kernel task summary. This matches the KE_NUM shown in the full trace entries for the task, and enables you to distinguish the CICS processing associated with the task you are interested in from other CICS processing.

**Finding more information about the error**

More information about the failure is given in the summary information for the task in error. This is given after the kernel task summary. It gives you a storage report
for the task, including registers and PSWs, and any data addressed by
the registers. 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. An example of such a storage
report is shown in Figure 2 in this case for a program check.

Look first in the dump for this header, which introduces the error report for the task:
**KE: KE DOMAIN ERROR TABLE**

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

**KE: ERROR NUMBER: 00000001**

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

Some kernel error data follows. If you want to find the format of this data (and, in
most cases, you will not need to), see the DFHKERRD section of the [CICS Data
Areas](#). The next thing of interest is the kernel's interpretation of what went wrong.
This includes the error code, the error type, the name of the program that was
running, and the offset within the program.

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

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

**==KE: KE DOMAIN ERROR TABLE**
**=KE: ERROR NUMBER: 00000001**
**KERRD 0397B950 KERNEL ERROR DATA**

**0000 F0C3F461 C12C5C1 01A00C4 00002EE C4C6BEE3 E2D74040 0403FC70 054D0B78 +0C4/AKEA.D.D.....DFHTSP .L....** 0397B950

(Data for offset 0020 to 0100 follows)

**0020 04D43DB4 03B2F140 039936A0 00000001 0006A000 8003CBB0 84D43A54 04D43A3E **+E.M....1 .R..............DM...M..** 0397B9A0

(Data for offset 0020 to 2300 follows)

**DATA AT PSW: 84D41F5E MODULE: DFHTSP OFFSET: 000622EE**

**DATA AT REGS: 84D41F5E MOD: DFHTSP OFFSET: 000622EE**

**DATA AT REGS**
**REG 0 F2000518**
**REG 6 F1000518**
**REG 0 F2000518**
**REG 6 F1000518**

**31-BIT DATA CANNOT BE ACCESSED**

**24-BIT DATA FOLLOWS:**

**SIMILAR DATA FOR CICS REGISTERS 1 TO 15 FOLLOWS**

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

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

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

PROGRAM QQQQQQQQ WAS IN CONTROL, BUT THE PSW WAS ELSEWHERE.

The program name (qqqqqqqq) reported is the program that owns the current kernel stack entry for the abending task. If this text appears, it may be possible to locate the failing program using the method described in [Using the linkage stack to identify the failing module” on page 45]. The failing program name and offset are also displayed in the section of the report immediately after the contents of the registers have been reported. The format of this information is:

DATA AT PSW: AAAAAAAA MODULE: PPPPPPPP OFFSET: XXXXXXXX

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

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

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

After the kernel’s interpretation of the error, you will see one of these diagnostic messages:

Error happened under the CICS RB

This means that 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 these circumstances:

- An error occurs in CICS SVC code.
- An error occurs in a CICS VTAM 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.

After either of these messages, you next get some data that is likely to be related to the problem. The data you get depends on whether or not the error happened under the CICS RB.

**The error data for the failing task**

If the error happened under the CICS RB, the error data you get in the task storage report is based on values in the PSW and the CICS registers at the time the error was detected. [Figure 2 on page 42] 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 may 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.

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

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

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

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

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

If, for any reason, the register does not address any data, you see either of these messages:
- 24-bit data cannot be accessed
- 31-bit data cannot be accessed

This means that the addresses cannot be found in the system dump of the CICS region. Note that 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 within the CICS region, but they were not included in the SDUMP. This is because MVS enables you to take SDUMPs selectively, for example “without LPA”. If this were to happen without your knowledge, you might think you had an addressing error when, in fact, the address was a valid one.

The format of the PSW is described in the IBM Enterprise Systems Architecture/370 Principles of Operation. The information in the PSW can help you to find the details needed by the IBM Support Center. You can find the address of the failing instruction, and hence its offset within the module, and also the abend type. You
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."

**Using the linkage stack to identify the failing module**

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

This method is only valid if the abend has occurred in a module or subroutine that
has a kernel linkage stack entry. This is the case only where the module or
subroutine has been invoked by one of these mechanisms:
- A kernel domain call
- A kernel subroutine call
- A call to an internal procedure identified to the kernel
- A LIFO call

Routines that have been invoked by assembler language BALR instructions do not
have kernel linkage stack entries. Having found which task was in error from the
kernel's task summary (see "Finding which tasks are associated with the error" on
page 40), you need next to find out which module was in error. The module name is
one of the things you need to give the IBM Support Center when you report the
problem to them.

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

Figure 3 shows a typical kernel linkage stack:

<table>
<thead>
<tr>
<th>KE_NUM</th>
<th>0STACK</th>
<th>LEN</th>
<th>TYPE</th>
<th>ADDRESS</th>
<th>LINK</th>
<th>REG</th>
<th>OFFS</th>
<th>ERROR</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0031</td>
<td>0520A020 0120 Bot 84C00400 84C00600 0200</td>
<td>DFHKEA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520A140 01F0 Dom 84C0F078 84C0F18E 0116</td>
<td>DFHDSKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520A330 0370 Dom 84CAAA8A 84CAAC20 071A</td>
<td>DFHXMTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520A6A0 0330 Dom 84F25430 84F25CF6 08C6</td>
<td>DFHPGPG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+00CC 84F254B6 0068</td>
<td>INITIAL_LINK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520A9D0 03C0 Dom 84F6C230 84E5DC40 0000</td>
<td>DFHAP11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+0EEA 84F6C66E 043E</td>
<td>CICS_INTERFACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520AD90 0108 Sub 8230B400 8230B680 04CA</td>
<td>DFHEQSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520AE9B 0290 Sub 82136D90 82137178 03EB +YES+ DFHLLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0520B128</td>
<td>Int +0BFC 82136F26 0196</td>
<td>LDL_INQUIRE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0520B128</td>
<td>Int +12BE 821376CE 093E</td>
<td>CURRENT_GET_NO_WAIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520B128 0F70 Dom 84C6FBE0 84C72EA6 35C6</td>
<td>DFHMEME_ _ _</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+2C86 84C6FAAE 016E</td>
<td>SEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+1486 84C726B4 0DA4</td>
<td>CONTINUE_SEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+350E 84C70DE4 1504</td>
<td>TAKE_A_DUMP_FOR_CALLER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>0520C098 0310 Dom 84C52458 84C52F52 0AFF</td>
<td>DFHDDDD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+0FB4 84C5254A 00F2</td>
<td>SYSTEM_DUMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>+1412 84C53212 00BA</td>
<td>TAKE_SYSTEM_DUMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

The TYPE column in the example can contain any of the following entries:
- **Bot** This marks the first entry in the stack.
- **Dom** This marks a stack entry caused by a domain call.
- **Sub** This marks a stack entry caused by a subroutine.
- **Lifo** This marks a stack entry caused by a LIFO module.
This marks a call to an internal procedure identified to the kernel.

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

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

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

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

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

Using the PSW to find the offset of the failing instruction

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

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

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

Finding the PTF level of the module in error

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

Figure 4 on page 47 shows some entries from a typical 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 – (dash) symbol appears in the appropriate column. If AMODE_ANY or RMODE_ANY is specified, ‘ANY’ appears in the appropriate column. Other values are shown as specified.

<table>
<thead>
<tr>
<th>PGM NAME</th>
<th>ENTRY PT</th>
<th>CSECT</th>
<th>LOAD PT. REL.</th>
<th>PTF LVL.</th>
<th>LAST COMPILED</th>
<th>COPY NO.</th>
<th>USERS</th>
<th>LOCN</th>
<th>TYCP</th>
<th>ATTRIBUTE</th>
<th>R/A MODE</th>
<th>APE ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCSA</td>
<td>80049810</td>
<td>DFHKELCL</td>
<td>00049000 520</td>
<td>12/11/96</td>
<td>10.09</td>
<td>1</td>
<td>1</td>
<td>CDSA</td>
<td>RPL</td>
<td>RESIDENT</td>
<td>-</td>
<td>07F16730</td>
</tr>
<tr>
<td>DFHKELRT</td>
<td>00049380</td>
<td>520</td>
<td>12/11/96 10.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHCDSA</td>
<td>00049608</td>
<td>0520</td>
<td>01/04 14.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHCSDOF</td>
<td>000496A8</td>
<td>0520</td>
<td>01/04 14.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHKERCD</td>
<td>00049FE0</td>
<td>520</td>
<td>11/14/96 18.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHKERER</td>
<td>0004A1B8</td>
<td>520</td>
<td>11/14/96 18.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHKESFM</td>
<td>0004AF58</td>
<td>520</td>
<td>12/11/96 10.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHKESGM</td>
<td>0004B830</td>
<td>510</td>
<td>10/21/96 16.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCP</td>
<td>8004D020</td>
<td>DFHTCP</td>
<td>0004D000 0520</td>
<td>11/12/10.10</td>
<td>1</td>
<td>1</td>
<td></td>
<td>CDSA</td>
<td>RPL</td>
<td>RESIDENT</td>
<td>-</td>
<td>07F5C430</td>
</tr>
<tr>
<td>DFHTCORS</td>
<td>0004D028</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCCOM</td>
<td>0004D538</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCSSS</td>
<td>0004D8B0</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCTTI</td>
<td>0004DA28</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCSAM</td>
<td>0004DA80</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCAM</td>
<td>0004EB38</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCTRN</td>
<td>0004ED30</td>
<td>0520</td>
<td>11/12/10.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHTCTDY</td>
<td>0004FA30</td>
<td>DFHTCTDY</td>
<td>0004FA10 0520</td>
<td>04/02 19.16</td>
<td>1</td>
<td>1</td>
<td></td>
<td>CDSA</td>
<td>RPL</td>
<td>RESIDENT</td>
<td>-</td>
<td>07F5C530</td>
</tr>
</tbody>
</table>

*Figure 4. Part of the loader domain program storage map summary*
Chapter 6. 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 during:

- A run
- Initialization
- Termination
- Restart

It contains the following topics:

- “Techniques for investigating waits” on page 51
- “Investigating terminal waits” on page 57
- “Investigating storage waits” on page 67
- “Investigating temporary storage waits” on page 69
- “Investigating enqueue waits” on page 72
- “Investigating interval control waits” on page 75
- “Investigating file control waits” on page 80
- “Investigating loader waits” on page 93
- “Investigating lock manager waits” on page 93
- “Resolving deadlocks in a CICS region” on page 101
- “Resolving deadlocks in a sysplex” on page 105
- “Resolving in-doubt and resynchronization failures” on page 106
- “What to do if CICS has stalled” on page 106

If CICS has stalled, turn directly to “What to do if CICS has stalled” on page 106.

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 7.

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 Chapter 9, “Dealing with performance problems,” on page 161.

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 140.
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 and/or the resource name identifying the unavailable resource. CEMT INQ TASK displays the resource type of the unavailable resource in the HTYPE field. The HVALUE field displays the resource name of the unavailable resource. EXEC CICS INQUIRE TASK returns values in the SUSPENDTYPE and SUSPENDVALUE fields which correspond to the resource type and resource name of the unavailable resource.

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

Table 9 on page 113 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,

**Setting up trace for wait problems**

Gate DSSR of dispatcher domain provides the major functions associated with the
suspension and resumption of tasks. (See Chapter 7, “How tasks are made to wait,”
on page 111.) The level-1 trace points DS 0004 and DS 0005 are produced on
entry to, and exit from, the gate.

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

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

**Interpreting trace for 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.

Interpreting the dump for wait problems

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

### Table 3. Descriptions of fields shown in the dispatcher task summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>The 2-character domain index identifying the domain that attached the task to the dispatcher.</td>
</tr>
<tr>
<td>ATTACHER TOKEN</td>
<td>A token provided by the domain that attached the task. This token uniquely identifies the task to the attaching domain.</td>
</tr>
<tr>
<td>DS_TOKEN</td>
<td>A token given by the dispatcher to a domain that attaches a task. It identifies the attached task uniquely to the dispatcher.</td>
</tr>
<tr>
<td>DTA</td>
<td>An address used internally by the dispatcher.</td>
</tr>
<tr>
<td>KE_TASK</td>
<td>A value that uniquely identifies to the kernel a task that has been created.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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 time-out, CEMT SET TASK PURGE, and EXEC CICS SET TASK PURGE.  
            | The possible values are:  
            | Y (=YES)—the task is purgeable.  
            | N (=NO)—the task is not purgeable. |
| 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 Chapter 7, “How tasks are made to wait,” on page 111.  
            | 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.) |
| ST (suspended tasks only) | The type of function that was invoked to suspend a currently suspended task. Possible values include:  
            | MVS—function WAIT_MVS  
            | OLD—function WAIT_OLDC  
            | OLWD—function WAIT_OLDW  
            | SUSP—function SUSPEND  
            | For a description of the functions, see Chapter 7, “How tasks are made to wait,” on page 111. |
Table 3. Descriptions of fields shown in the dispatcher task summary (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSPAREA (suspended</td>
<td>Either an address used internally by the dispatcher, or an ECB address, or an ECB list address. These are the cases:</td>
</tr>
<tr>
<td>tasks only)</td>
<td>• Address used internally, if the task was suspended by a SUSPEND call.</td>
</tr>
<tr>
<td></td>
<td>• ECB address or ECB list address, if the task was suspended by a WAIT_MVS or WAIT_OLDW call.</td>
</tr>
<tr>
<td></td>
<td>• ECB address, if the task was suspended by a WAIT_OLDC call.</td>
</tr>
<tr>
<td></td>
<td>Look at the value given in the ST column to see which one of these descriptions applies.</td>
</tr>
<tr>
<td>TIME OF SUSPEND</td>
<td>The time when a currently suspended task was suspended.</td>
</tr>
<tr>
<td>(suspended tasks only)</td>
<td>The format is hh:mm:ss.mmm (hours, minutes, seconds, milliseconds), GMT.</td>
</tr>
<tr>
<td>TIMEOUT DUE</td>
<td>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.</td>
</tr>
<tr>
<td>(suspended tasks only)</td>
<td>The format is hh:mm:ss.mmm (hours, minutes, seconds, milliseconds).</td>
</tr>
<tr>
<td>TT</td>
<td>The time-out type for the task. The possible values, where one is given, are:</td>
</tr>
<tr>
<td>(suspended tasks only)</td>
<td>IN—a time-out interval has been specified for the task.</td>
</tr>
<tr>
<td></td>
<td>DD—deadlock action is to be delayed when the time-out interval expires.</td>
</tr>
<tr>
<td></td>
<td>DI—deadlock action is immediate when the time-out interval expires.</td>
</tr>
<tr>
<td>TY</td>
<td>Whether this is a system task or a non-system task. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• SY—this is a system task.</td>
</tr>
<tr>
<td></td>
<td>• NS—this is a non-system task.</td>
</tr>
<tr>
<td></td>
<td>A non-system task can be either a user written transaction, or a CICS-supplied transaction.</td>
</tr>
</tbody>
</table>

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 52.

Table 4 on page 56 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 4. Parameters and functions that set fields shown in the dispatcher task summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Related parameter</th>
<th>Function</th>
<th>Input or output</th>
<th>Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>DOMAIN_INDEX</td>
<td>INQUIRE_TASK</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTA</td>
<td>ATTACH_TOKEN</td>
<td>CREATE_TASK</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td>DS_TOKEN</td>
<td>TASK_TOKEN</td>
<td>ATTACH</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL_TASK</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PURGE_INHIBIT_QUERY</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PRIORITY</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TASK_REPLY</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>KE_TASK</td>
<td>TASK_TOKEN</td>
<td>CREATE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CREATE_TCB</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUSH_TASK</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TASK_REPLY</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCB_REPLY</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td>MO</td>
<td>MODE</td>
<td>ATTACH</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHANGE_MODE</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>PURGEABLE</td>
<td>SUSPEND</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_MVS</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDC</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDW</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESOURCE NAME</td>
<td>RESOURCE_NAME</td>
<td>ADD_SUSPEND</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSPEND</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_MVS</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDC</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDW</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>RESOURCE TYPE</td>
<td>RESOURCE_TYPE</td>
<td>ADD_SUSPEND</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSPEND</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_MVS</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDC</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDW</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>STATE</td>
<td>GET_NEXT</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TASK</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>SUSPAREA</td>
<td>ECB_ADDRESS or ECB_LIST_ADDRESS</td>
<td>WAIT_MVS</td>
<td>IN</td>
<td>DSSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDC</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_OLDW</td>
<td>IN</td>
<td></td>
</tr>
<tr>
<td>TIME OF SUSPEND</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>TASKNO</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>TIMEOUT DUE</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>
### Investigating terminal waits

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

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

Note that, if you have one or more unresponsive terminals, that is terminals that are showing no new output and accepting no input, this does not necessarily indicate a terminal wait. If you have this problem, use CEMT INQ TERMINAL to find the transaction running at the terminal, and then CEMT INQ TASK to find out what resource that task is waiting on. When you know that, look at Table 9 on page 113 to find where you can get further guidance.

If all the terminals in the network are affected, and CICS has stalled, read “What to do if CICS has stalled” on page 106 for advice about how to investigate the problem.

If yours is a genuine terminal wait, remember when you carry out your investigation that terminals in the CICS environment can have a wide range of characteristics. A terminal is, in fact, anything that can be at the end of a communications line. It could, for example, be a physical device such as a 3270 terminal or a printer, 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

Before taking a systematic approach to the problem, here are a few preliminary considerations that could point to a simple solution.

- Is there an obvious physical explanation for the wait? For example, is a terminal operator failing to respond to a request for input? In the case of a printer, has it been powered off, or has it run out of paper?

- Have you checked in the CSTL and CSNE logs to see if there is a message you might have missed? If either DFHTCP or DFHZCP detected an error related to terminal control, a message reporting the problem will have been sent to the CSNE log, and, perhaps, to the console too.

If you do find a message there reporting some terminal error that can be related to the task, it should give you an idea of why the task is waiting. You can find an explanation of the message and a description of the system action in response to the error by using the CMAC transaction, or by looking at CICS Messages and Codes.

It is also possible that the CSNE log entry will show that an error was detected, but that no action was taken by TACP or NACP. This could occur if the line or terminal was out of service, or if the error actions had been turned off in the user exits of DFHTEP and DFHNEP. In such a case, the CICS code enabling the waiting task to be resumed would never be executed, and the task would wait indefinitely.

- Have any HANDLE CONDITION routines for terminal errors been coded incorrectly? If an attempt were made to access the terminal having the error in such a routine, the application would be very likely to wait indefinitely.

- If the terminal is installed using autoinstall, has the system failed to load DFHZATA, the autoinstall program, or DFHZCQ which is called by DFHZATA, because of a short on storage condition? If so, you need to deal with the cause of the short on storage condition. Check the delete delay that you have specified—if it is too short, your system may be deleting and reinstalling terminals unnecessarily. If storage fragmentation is preventing DFHZATA or DFHZCQ from being loaded consider defining them as resident. However, be aware that DFHZCQ is a large program, and check your storage requirements before making this change.

If none of these considerations applies, you need to 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 identified in the *CICS Resource Definition Guide*.

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

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 TCTTEAMIB, which is the byte name definition for the access method. The *CICS Data Areas* relates values to access methods.

The most common access method is VTAM, identified by a value of TCTEVTAM. The problem determination procedures outlined below focus exclusively on VTAM. 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 66.
VTAM in use—debugging procedures

The sections that follow give guidance about debugging terminal waits when the access method is VTAM.

The first step is to look in the CSNE log to see if there is an error message that explains the wait. If it contains an error code, you can find out what it means from the VTAM Messages and Codes for MVS and VSE manual.

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

Is the problem associated with VTAM?

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCTECIP</td>
<td>command request in progress</td>
</tr>
<tr>
<td>TCTEDIP</td>
<td>data request in progress</td>
</tr>
</tbody>
</table>

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

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

**VTAM submodule identifiers:** This table contains Product-sensitive Programming Interface information.

<table>
<thead>
<tr>
<th>Hex ID</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'00'</td>
<td>ZDSP</td>
<td>DISPATCH</td>
</tr>
<tr>
<td>X'01'</td>
<td>ZARQ</td>
<td>READ /WRITE R</td>
</tr>
<tr>
<td>X'02'</td>
<td>ZLOC</td>
<td>LOCATE</td>
</tr>
<tr>
<td>X'03'</td>
<td>ZDET</td>
<td>DETACH</td>
</tr>
<tr>
<td>X'04'</td>
<td>ZTCP</td>
<td>TCP</td>
</tr>
<tr>
<td>X'06'</td>
<td>ZCRQ</td>
<td>COMMAND REQS</td>
</tr>
<tr>
<td>X'08'</td>
<td>ZSTU</td>
<td>STATUS CHANGE</td>
</tr>
<tr>
<td>X'09'</td>
<td>ZTSP</td>
<td>TERMINAL SHARING</td>
</tr>
<tr>
<td>X'0A'</td>
<td>ZHPX</td>
<td>HPO RPL EXEC OS ONLY</td>
</tr>
<tr>
<td>X'0B'</td>
<td>ZISP</td>
<td>ALLOCATE/FREE</td>
</tr>
<tr>
<td>Hex ID</td>
<td>Module</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>X'0C'</td>
<td>ZIS1</td>
<td>INTER SYSTEM</td>
</tr>
<tr>
<td>X'0D'</td>
<td>ZIS2</td>
<td>INTER SYSTEM 2</td>
</tr>
<tr>
<td>X'0E'</td>
<td>ZABD</td>
<td>INVALID REQUEST/ABEND</td>
</tr>
<tr>
<td>X'10'</td>
<td>ZATI</td>
<td>ATI</td>
</tr>
<tr>
<td>X'11'</td>
<td>ZATT</td>
<td>ATTACH TASK</td>
</tr>
<tr>
<td>X'12'</td>
<td>ZFRE</td>
<td>FREE STORAGE</td>
</tr>
<tr>
<td>X'13'</td>
<td>ZGET</td>
<td>GET STORAGE</td>
</tr>
<tr>
<td>X'14'</td>
<td>ZRAC</td>
<td>RECEIVE ANY</td>
</tr>
<tr>
<td>X'15'</td>
<td>ZRST</td>
<td>RESETSR</td>
</tr>
<tr>
<td>X'16'</td>
<td>ZRVS</td>
<td>RECEIVE SPEC</td>
</tr>
<tr>
<td>X'17'</td>
<td>ZRVX</td>
<td>RECEIVE S EXT</td>
</tr>
<tr>
<td>X'18'</td>
<td>ZSDS</td>
<td>SEND NORMAL</td>
</tr>
<tr>
<td>X'19'</td>
<td>ZSDX</td>
<td>SEND DATA EXIT</td>
</tr>
<tr>
<td>X'1A'</td>
<td>ZUCT</td>
<td>TRANSLATION</td>
</tr>
<tr>
<td>X'1B'</td>
<td>ZUIX</td>
<td>USER EXIT</td>
</tr>
<tr>
<td>X'1C'</td>
<td>ZACT</td>
<td>ACTIVATE SCAN</td>
</tr>
<tr>
<td>X'1D'</td>
<td>ZSDR</td>
<td>SEND RESPONSE</td>
</tr>
<tr>
<td>X'1E'</td>
<td>ZHPS</td>
<td>HPO SEND/RECV CALL</td>
</tr>
<tr>
<td>X'1F'</td>
<td>ZRPL</td>
<td>RECV.ANY BLDER</td>
</tr>
<tr>
<td>X'20'</td>
<td>ZAIT</td>
<td>ATTACH INIT</td>
</tr>
<tr>
<td>X'21'</td>
<td>ZASX</td>
<td>ASYN COM EXIT</td>
</tr>
<tr>
<td>X'22'</td>
<td>ZCLS</td>
<td>CLOSE DESTIN</td>
</tr>
<tr>
<td>X'23'</td>
<td>ZCLX</td>
<td>CLOSE DS EXIT</td>
</tr>
<tr>
<td>X'24'</td>
<td>ZDWE</td>
<td>DWE PROCESS</td>
</tr>
<tr>
<td>X'25'</td>
<td>ZLEX</td>
<td>LERAD EXIT</td>
</tr>
<tr>
<td>X'26'</td>
<td>ZLGX</td>
<td>LOGON EXIT</td>
</tr>
<tr>
<td>X'27'</td>
<td>ZLRP</td>
<td>LOGICAL REC</td>
</tr>
<tr>
<td>X'28'</td>
<td>ZLTX</td>
<td>LOSTERM EXIT</td>
</tr>
<tr>
<td>X'29'</td>
<td>ZOPN</td>
<td>OPEN DESTINAT</td>
</tr>
<tr>
<td>X'2A'</td>
<td>ZOPX</td>
<td>OPEN DESTEIX</td>
</tr>
<tr>
<td>X'2B'</td>
<td>ZRAQ</td>
<td>READAHEAD QUE</td>
</tr>
<tr>
<td>X'2C'</td>
<td>ZRAR</td>
<td>READAHEAD RET</td>
</tr>
<tr>
<td>X'2E'</td>
<td>ZRRX</td>
<td>REL REQUEST EX</td>
</tr>
<tr>
<td>X'2F'</td>
<td>ZNSP</td>
<td>NETWORK SPEC EXIT</td>
</tr>
<tr>
<td>X'30'</td>
<td>ZRSY</td>
<td>RESYNC</td>
</tr>
<tr>
<td>X'31'</td>
<td>ZSAX</td>
<td>SEND COMM EXT</td>
</tr>
<tr>
<td>X'32'</td>
<td>ZSCX</td>
<td>SCIP EXIT</td>
</tr>
<tr>
<td>X'33'</td>
<td>ZSDA</td>
<td>SEND ASYN COM</td>
</tr>
<tr>
<td>X'34'</td>
<td>ZSKR</td>
<td>SEND COMMAND RESPONSE ID</td>
</tr>
<tr>
<td>X'35'</td>
<td>ZSES</td>
<td>SESSIONC COM</td>
</tr>
<tr>
<td>X'36'</td>
<td>ZSEX</td>
<td>SESSIONC EXIT</td>
</tr>
<tr>
<td>X'37'</td>
<td>ZSIM</td>
<td>SIMLOGON</td>
</tr>
<tr>
<td>X'38'</td>
<td>ZSIX</td>
<td>SIMLOGON EXIT</td>
</tr>
<tr>
<td>X'39'</td>
<td>ZSLS</td>
<td>SETLOGON START</td>
</tr>
<tr>
<td>X'3A'</td>
<td>ZSSX</td>
<td>SEND COM EXIT</td>
</tr>
<tr>
<td>X'3B'</td>
<td>ZSYX</td>
<td>SYNAAD EXIT</td>
</tr>
<tr>
<td>X'3C'</td>
<td>ZTAX</td>
<td>TURNAROUND EXIT</td>
</tr>
<tr>
<td>X'3D'</td>
<td>ZTPX</td>
<td>TPEND EXIT</td>
</tr>
<tr>
<td>X'3E'</td>
<td>ZOPA</td>
<td>VTAM OPEN ACB</td>
</tr>
<tr>
<td>X'3F'</td>
<td>ZSHU</td>
<td>VTAM SHUTDOWN</td>
</tr>
<tr>
<td>X'40'</td>
<td>ZQUE</td>
<td>TERMINAL SHARING</td>
</tr>
<tr>
<td>X'41'</td>
<td>ZEMW</td>
<td>ERROR MESSAGE WRITER</td>
</tr>
<tr>
<td>X'42'</td>
<td>ZSYN</td>
<td>SYNCPOINT HANDLER</td>
</tr>
</tbody>
</table>

Chapter 6. Dealing with waits  61
<table>
<thead>
<tr>
<th>Hex ID</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'43'</td>
<td>ZTRA</td>
<td>VTAM RPL TRACE</td>
</tr>
<tr>
<td>X'44'</td>
<td>ZAND</td>
<td>ABEND CONTROL BLOCK</td>
</tr>
<tr>
<td>X'45'</td>
<td>ZCNA</td>
<td>CONSOLE CONTROL</td>
</tr>
<tr>
<td>X'46'</td>
<td>ZCNR</td>
<td>CONSOLE REQUEST</td>
</tr>
<tr>
<td>X'47'</td>
<td>ZCNC</td>
<td>CONSOLE ABNORMAL COND.</td>
</tr>
<tr>
<td>X'48'</td>
<td>ZUAX</td>
<td>ATTACH USER EXIT</td>
</tr>
<tr>
<td>X'49'</td>
<td>ZUOX</td>
<td>OUTPUT USER EXIT</td>
</tr>
<tr>
<td>X'4A'</td>
<td>ZARL</td>
<td>LU6.2 APPL REQUEST</td>
</tr>
<tr>
<td>X'4B'</td>
<td>ZARM</td>
<td>LU6.2 MIGRATION</td>
</tr>
<tr>
<td>X'4C'</td>
<td>ZRLU</td>
<td>LU6.2 RECEIVE</td>
</tr>
<tr>
<td>X'4D'</td>
<td>ZRLX</td>
<td>LU6.2 RECEIVE EXIT</td>
</tr>
<tr>
<td>X'4E'</td>
<td>ZSDL</td>
<td>LU6.2 SEND</td>
</tr>
<tr>
<td>X'4F'</td>
<td>ZSLX</td>
<td>LU6.2 SEND EXIT</td>
</tr>
<tr>
<td>X'50'</td>
<td>ZERH</td>
<td>LU6.2 APPL ERP</td>
</tr>
<tr>
<td>X'52'</td>
<td>ZBKT</td>
<td>LU6.2 BRACKET STATE M/C</td>
</tr>
<tr>
<td>X'53'</td>
<td>ZCNT</td>
<td>LU6.2 CONTENTION STATE</td>
</tr>
<tr>
<td>X'54'</td>
<td>ZCHS</td>
<td>LU6.2 CHAIN SEND</td>
</tr>
<tr>
<td>X'55'</td>
<td>ZCHR</td>
<td>LU6.2 CHAIN RECEIVE</td>
</tr>
<tr>
<td>X'56'</td>
<td>ZUSR</td>
<td>LU6.2 CONVERSATION STATE</td>
</tr>
<tr>
<td>X'57'</td>
<td>ZDST</td>
<td>SNA-ASCII TRAN ROUTINE</td>
</tr>
<tr>
<td>X'58'</td>
<td>ZEV1</td>
<td>ENCRYPTION VALIDATION 1</td>
</tr>
<tr>
<td>X'59'</td>
<td>ZEV2</td>
<td>ENCRYPTION VALIDATION 2</td>
</tr>
<tr>
<td>X'5E'</td>
<td>ZXRC</td>
<td>XRF TERMINAL RECOVERY</td>
</tr>
<tr>
<td>X'5F'</td>
<td>ZXTS</td>
<td>XRF TERMINAL SCAN</td>
</tr>
<tr>
<td>X'60'</td>
<td>ZXRl</td>
<td>LU6.2 Transaction Routing</td>
</tr>
<tr>
<td>X'61'</td>
<td>ZINT</td>
<td>Initialization Module Ident</td>
</tr>
<tr>
<td>X'62'</td>
<td>ZXRT</td>
<td>LU6.2 Transaction Routing TOS</td>
</tr>
<tr>
<td>X'63'</td>
<td>ZSTA</td>
<td>LU6.2 Application Status</td>
</tr>
<tr>
<td>X'64'</td>
<td>ZRLP</td>
<td>LU6.2 RECEIVE post-vtam</td>
</tr>
<tr>
<td>X'65'</td>
<td>ZCRT</td>
<td>LU6.2 RPL_B state</td>
</tr>
<tr>
<td>X'66'</td>
<td>ZRAS</td>
<td>LU6.2 Slow-down processing</td>
</tr>
<tr>
<td>X'67'</td>
<td>ZXPS</td>
<td>LU6.2 Per sess recovery</td>
</tr>
<tr>
<td>X'7D'</td>
<td>ZRLG</td>
<td>RESPONSE LOGGER</td>
</tr>
<tr>
<td>X'7E'</td>
<td>ZNAC</td>
<td>NACP</td>
</tr>
<tr>
<td>X'7F'</td>
<td>ZRSP</td>
<td>RESYNC SYSTEM TASK</td>
</tr>
<tr>
<td>X'80'</td>
<td>ZATR</td>
<td>ZATR restart deletes</td>
</tr>
<tr>
<td>X'82'</td>
<td>ZATA</td>
<td>ZATA autoinstall</td>
</tr>
<tr>
<td>X'84'</td>
<td>ZATD</td>
<td>ZATD autoinstall delete</td>
</tr>
<tr>
<td>X'86'</td>
<td>ZGMM</td>
<td>GOOD MORNING TRANSACTION</td>
</tr>
<tr>
<td>X'8B'</td>
<td>ZATS</td>
<td>ZATS remote install entry</td>
</tr>
<tr>
<td>X'C0'</td>
<td>ZQ00</td>
<td>DFHZCQ REQUEST ROUTER</td>
</tr>
<tr>
<td>X'C1'</td>
<td>ZQIN</td>
<td>ZC INITIALIZE</td>
</tr>
<tr>
<td>X'C2'</td>
<td>ZQBA</td>
<td>ZC Bind Analysis</td>
</tr>
<tr>
<td>X'C3'</td>
<td>ZQCH</td>
<td>ZC CHANGE</td>
</tr>
<tr>
<td>X'C4'</td>
<td>ZQDL</td>
<td>ZC DELETE</td>
</tr>
<tr>
<td>X'C5'</td>
<td>ZQIT</td>
<td>ZC INSTALL TCTTE</td>
</tr>
<tr>
<td>X'C6'</td>
<td>ZQRC</td>
<td>ZC RECOVER</td>
</tr>
<tr>
<td>X'C7'</td>
<td>ZQRS</td>
<td>ZC RESTORE</td>
</tr>
<tr>
<td>X'C8'</td>
<td>ZQIQ</td>
<td>ZC INQUIRE</td>
</tr>
</tbody>
</table>
If you suspect that the problem is associated with VTAM, consider using either CICS VTAM exit tracing or VTAM buffer tracing. Both of these techniques can give you detailed information about the execution of VTAM requests. For guidance about using the techniques, read the appropriate sections in Chapter 15, “Using traces in problem determination,” on page 225.

**VTAM in use—is terminal 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.

**VTAM in use—is the terminal at fault?**

If you have found that the access method and terminal control are not causing the wait, the terminal itself must be waiting for some reason. 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 VTAM terminal entries. It appears in the terminal control (TCP) summary section of the dump.

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

The example in Figure 6 on page 64 shows the index followed by a terminal entry with its interpreted status and attribute information.
The values that are given below for fields in the TCTTE are not the only possibilities, but they show important things about the terminal status. If you find any other values for these fields, look in the CICS Data Areas 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 VTAM from field TCTEIMOST, and, if bracket protocol is required (from field TCTEIBPE), its conversation state with the terminal from field TCTEINB. 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 TCTEINB:

   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 TCTELOS, TCTEOPD, and TCTENSQ:

   TCTELOS = 1 the node is logged on
   TCTEOPD = 1 VTAM OPNDST macro issued
   TCTENSQ = 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:

   TCTENNO = 1 the terminal is to be logged off
   TCTENBO = 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:

   TCTEIPRA = 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 VTAM is in use**

Amongst your debugging tools, two are likely to be of particular use for investigating terminal waits in a VTAM environment. They are:

- VTAM buffer trace. This is a feature of VTAM itself, and you need to see the appropriate manual in the VTAM library for details of how to use it.
- CICS VTAM exit trace. This is a feature 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 Chapter 15, "Using traces in problem determination," on page 225.

**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.

**Your task is waiting on a physical 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 9 on page 113 to find out which part of this section to turn to next.

**Investigating the related task in the remote region**

The first thing to do is to 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.

When you have identified the region, you need to 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. Take a system dump of the local region, too.
Format the dumps using the formatting keyword DS to get a summary of the tasks in each region, and TCP to get the TCTTEs.

First find the TCTTE for the task in the local 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.

When you have confirmed that you have located the correct TCTTE, look in field TCTTECA. This gives you the TCA address of the task that is not responding.

Using the TCA address as the entry point, you can now investigate the reason why the task has not responded. It is very likely that it has been suspended because some resource is unavailable. Look in the dispatcher and transaction manager summaries. If you can identify your task, you can see what resource it is waiting on.

When you have identified the resource, turn to the appropriate section in this section for guidance about investigating waits on that resource.

## Investigating storage waits

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

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

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

In addition, your task may be automatically purged if it has waited for storage longer than the deadlock time-out parameter specified in the installed transaction definition. Certain conditions prevent purging of a task, for example, a deadlock time-out value of 0, or a specification of SPURGE(NO). The two most likely explanations for extended waits on storage requests are:
1. The task has issued an unconditional GETMAIN request for an unreasonably large amount of storage.

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

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

**Was the request for too much storage?**

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

```
==SM: Suspend queue summary
KE Task Tran # Susptok Subpool DSA Request
053E5400 0000041 04080011 U0000041 EUDSA 10000016
```

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

**Is the storage close to being exhausted?**

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

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

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

If the amount of free space is unexpectedly small, look at the task subpool summary. If a task has made an unusually large number of GETMAIN requests, this could indicate that it is looping. A looping task might be issuing GETMAIN requests repetitively, each for a reasonable amount of storage, but collectively for a very large amount. If you find evidence for a looping task, turn to Chapter 8, “Dealing with loops,” on page 143.

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

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

Investigating temporary storage waits

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

Resource type ENQUEUE

Because enqueues on temporary storage should be held in retained state, no user task should ever wait on a resource type of ENQUEUE with a value of TSNQ (temporary storage enqueue).

For general information about dealing with enqueue waits, see "Investigating enqueue waits" on page 72.

Resource type TSAUX

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

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 first step is to get a CICS system dump, and format it using the formatting keyword TS to show the temporary storage control blocks. Include formatting keywords SM and KE, too, as you might need to refer to the summaries for these two components as well. The way you analyze the dump to investigate the cause of the problem is described in the sections that follow.

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 Chapter 8, “Dealing with loops,” on page 143.

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 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

  \[ \text{(CIs in data set - CIs in use)} < \text{(record length / available bytes per CI)} \]

- No control interval has enough contiguous space to accommodate the remainder.

**Resource type TSBUFFER**

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**

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

**Resource type TSIO**

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

**Resource type TSPOOL**

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**

Resource type TSQUEUE indicates that the waiting task has issued a request against a temporary storage queue that is already in use by another task. The latter task is said to have the lock on the queue.

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**

Resource type TSSHARED indicates that the task attempted to write to a shared temporary storage queue on the coupling facility, and there was insufficient space to satisfy the request. The request is retried every half second.

**Resource type TSSTRING**

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

**Resource type TSWBUFFR**

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

**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). Do not use the EXEC CICS ENQ command for recoverable resources.

There are two ways in which you can discover the owner of the enqueue that the task is waiting on:

- Use the CEMT INQUIRE UOWENQ command. For an example of how to use this command to discover the owner of an enqueue, see [101]. For definitive information about CEMT INQUIRE UOWENQ, see *CICS Supplied Transactions*.

- Use the NQ section of a system dump. 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"). The following resource names identify the enqueue waits that this applies to:
  - JOURNALS
  - KCADDR
  - KCSTRING
  - LOGSTRMS

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 [101].

**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 5 shows the resources that this applies to.

<table>
<thead>
<tr>
<th>Resource name</th>
<th>Type of resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOURNALS</td>
<td>CICS journal names used during creation, deletion, or use of a journal entry. See &quot;Log manager waits&quot; on page 129 for information to help you diagnose problems with the MVS system logger.</td>
</tr>
<tr>
<td>KCADDR</td>
<td>Addresses locked internally by CICS.</td>
</tr>
<tr>
<td>KCSTRING</td>
<td>Strings locked internally by CICS.</td>
</tr>
<tr>
<td>LOGSTRMS</td>
<td>MVS logstream names used during connection of streams to the MVS logger. A long wait could indicate a problem with the logger. See &quot;Log manager waits&quot; on page 129 for information to help you diagnose problems with the MVS system logger.</td>
</tr>
</tbody>
</table>

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
Figure 7 shows an example summary for the transient data enqueue (TDNQ) pool.

**NOTE:** These values were reset at 15.44.39 (the last statistics interval collection)

<table>
<thead>
<tr>
<th>Enqueue Name</th>
<th>Len</th>
<th>Sta</th>
<th>Address</th>
<th>Tran</th>
<th>Tran Id</th>
<th>Local Uowid</th>
<th>Uow Tsk</th>
<th>Lifetime</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q007TOQ</td>
<td>9</td>
<td>Act</td>
<td>052C4580</td>
<td>TDWR</td>
<td>00356</td>
<td>A8EBC70A53A4BC82</td>
<td>1 0 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q002FROMQ</td>
<td>9</td>
<td>Act</td>
<td>053D0880</td>
<td>TDRD</td>
<td>00435</td>
<td>A8EBO91A57D9670D2</td>
<td>2 0 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q007FROMQ</td>
<td>9</td>
<td>Act</td>
<td>0540CC80</td>
<td>ENQY</td>
<td>00217</td>
<td>A8EBA2AFDA4CD82</td>
<td>1 0 53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q002TOQ</td>
<td>9</td>
<td>Ret</td>
<td>0520B260</td>
<td>-----</td>
<td>-------</td>
<td>A8EBC84D3FF0250</td>
<td>1 0 62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q009FROMQ</td>
<td>9</td>
<td>Act</td>
<td>0540A140</td>
<td>TDRD</td>
<td>00366</td>
<td>A8EBC84D3FF0250</td>
<td>1 0 62</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

**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.
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 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. (For detailed information about the EXEC CICS ENQ command, see the [CICS Application Programming Reference](https://www.ibm.com/docs/en/zos-ics)).

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. (For detailed information about the EXEC CICS ENQ command, see the [CICS Application Programming Reference](https://www.ibm.com/docs/en/zos-ics)).

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. Note that, 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.
Investigating interval control waits

Read this section if you have a task that is not running, and interval control seems to be involved. The following is a list of possible 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, turn to “Finding the reason for a delay request not completing” on page 76 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 57.

- 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, for some reason, 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, for some reason, not available. The resource name gives you the name of the terminal running the task in the ICGTWAIT wait and therefore the target TERMD for other tasks issuing EXEC CICS START commands to supply more data. If there are no tasks in the system that would issue START commands for this TERMD, you need to determine whether this is reasonable. If there are such tasks in the system, check to see why they are not issuing the required START commands. They might, for example, be waiting for terminal input.

  Look, too, at the deadlock time-out interval (DTIMOUT) and the system purge value (SPURGE) for the task issuing the EXEC CICS RETRIEVE WAIT command. If there is no DTIMOUT value or SPURGE=NO has been specified, the task will wait indefinitely for the data.

  Note: The task waiting on resource ICGTWAIT might not be the one that you first set out to investigate. Any AID task scheduled to start at the same terminal cannot do so until the current task has terminated.

- You have found that the task is waiting on resource type ICWAIT. This means that the task issued an EXEC CICS DELAY command that has not yet completed. Check that the interval or time specified on the request was what you intended. If you believe that the expiry time of the request has passed, that suggests a possible CICS error.

  Consider, too, the possibility that the task was the subject of a long DELAY that was due to be canceled by some other task. If the second task failed before it could cancel the delay, the first would not continue until the full interval specified on DELAY had expired.

- A task that issued EXEC CICS POST did not have its ECB posted when you expected it to.

  Check to make sure the interval or time you specified was what you intended.

- A task that issued EXEC CICS WAIT EVENT was not resumed when you thought it should have been.

  Assuming the WAIT was issued sometime after a POST, first check to make sure that the interval or time specified on the POST was what you intended. If it was, next check to see whether the ECB being waited on was posted. If it was, that indicates a possible CICS error.
If none of the simple checks outlined here help you to solve the problem, read the next section.

Finding the reason for a DELAY request not completing

If your preliminary investigations have not shown the reason for the wait, you need to look in greater detail at the evidence available. Take a system dump, and format it using the keywords CSA, ICP, and AP. These 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.

First, locate field CSATODTU in the CSA. Make a note of the value there, which is the current CICS time of day in internal ‘timer units’. Now locate the TCA for your task, and read the value of field TCAICEAD. This gives you the address of the interval control element for your task. Use this information to find the ICE (interval control element) for the task, and look at field ICEXTOD. Make a note of the value there.

If ICEXTOD is greater than CSATODTU, the ICE has not yet reached the expiry time. These are the possible explanations:

- Your task either did not make the DELAY request you expected, or the interval specified was longer than intended. This could indicate a user error. Check the code of the transaction issuing the request to make sure it is correct.
- Your task’s delay request was not executed correctly. This might indicate an error within CICS code, or a corrupted control block.

If ICEXTOD is equal to CSATODTU (very unlikely), you probably took the system dump just as the interval was about to expire. In such a case, attempt to re-create the problem, take another system dump, and compare the values again.

If ICEXTOD is less than CSATODTU, the ICE has already expired. The associated task should have resumed. This indicates that some area of storage might have been corrupted, or there is an error within CICS code.

Using trace to find out why tasks are waiting on interval control

Before using trace to find out why your task is waiting on interval control, you need to select an appropriate trace destination and set up the right tracing options.

By their nature, interval control waits can be long, so select auxiliary trace as the destination, because you can specify large trace data sets for auxiliary trace. However, the data sets do not have to be large enough to record tracing for the whole interval specified when you first detected the problem. That is because the error is likely to be reproducible when you specify a shorter interval, if it is reproducible at all. For example, if the error was detected when an interval of 20 seconds was specified, try to reproduce it specifying an interval of 1 second.

As far as tracing selectivity is concerned, you need to capture level-2 trace entries made by dispatcher domain, timer domain, and interval control program. Use the CETR transaction to set up the following tracing options:

1. Specify special tracing for the level-2 trace points for components DS (dispatcher domain), TI (timer domain), and IC (interval control program).
2. Select special tracing for the task causing the problem, by specifying special tracing both for the transaction and for the terminal where it is to be run.
3. Set the master system trace flag off, to turn off all standard tracing. This helps minimize the number of trace entries not connected with the problem.

4. Make sure that auxiliary tracing is active, then set the transaction running. When the problem appears, format the auxiliary trace data set and either print it or view it online.

The sort of trace entries that you can expect in normal operation are shown in the figures that follow. They show the flow of data and control following execution of the command EXEC CICS DELAY INTERVAL(000003). A similar set of trace entries would be obtained if TIME had been specified instead of INTERVAL, because TIME values are converted to corresponding INTERVAL values before timer domain is called.

Figure 8 shows the first two entries that you get following execution of the EXEC CICS DELAY INTERVAL(000003) command.

Notes:

1. Trace point AP 00E1 is on ENTRY to the EIP DELAY routine. The function is stated in the trace header, and the fact that this trace is made on ENTRY can be deduced from the value shown in the request field, REQ(0004). The rightmost two bytes of FIELD B give the EIBFN value, in this case X'1004'. This shows that this is an interval control DELAY request.

   The value shown against TASK is the trace task number, and it is unique to the task while the task is in the system. Its purpose is to show which trace entries relate to which tasks. The task number in this example is 00163. As long as the task is in the system, and either running or suspended, trace entries having this task number always relate to it. Use the task number for your task to identify the trace entries associated with it.

2. Trace point AP 00F3 is on ENTRY to the ICP WAIT routine. The function is given explicitly in the trace header, and both the function and the fact that this represents ENTRY to the routine can be deduced from the request field, REQ(0003).

   The value of FIELD A, X'0000003C', is an important one for problem determination. It shows the interval that has been specified, in this case three seconds. Check the value shown here for your own task, to make sure it is what you expect it to be.

Look next for an entry with point ID DS 0004 showing your task being suspended, as in Figure 9 on page 78. 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.
Notes:

1. Trace point TI 0100, if shown, is on ENTRY to the REQUEST_NOTIFY_INTERVAL function of timer domain. This is stated explicitly in the trace header.

   The value shown in the header for STCK_INTERVAL is derived from the machine store clock value calculated for the DELAY interval specified on the EXEC CICS DELAY command. You can find out how store clock values are related to times in hours, minutes, and seconds from the ESA/370 Principles of Operation manual.

   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'00000002DC6C1000'. You can see that the actual store clock value is quoted in the trace entry as X'00000002DC6C1000', which is 3 seconds plus 1 microsecond.

   The TIME field of the trace entry shows the time at which the entry was made, in the format hh:mm:ss. The value in this example (ignoring the fractions of a second) is 16:31:58. It follows that the task is due to be resumed when the time is 16:32:01, because the interval is 3 seconds.

2. Trace point TI 0101, if shown, is on EXIT from the REQUEST_NOTIFY_INTERVAL function of timer domain.

   You can see from RESPONSE(OK) in the header that the function completed normally.

3. Trace point DS 0004 is on ENTRY to the dispatcher task SUSPEND/RESUME interface.

   The SUSPEND_TOKEN field in the trace header is significant. It shows the unique suspend token being used for this SUSPEND/RESUME dialog, and it is referred to explicitly again in a later trace entry showing that the task has been resumed. In this example, the suspend token is X'01040034'.

   Any subsequent dispatcher trace entry that shows the suspend token for your task is connected with the suspension or resumption of the task.

   Field RESOURCE_TYPE(ICWAIT) in the trace header shows that the resource type associated with this suspend is ICWAIT. ICWAIT is the resource type that is returned on CEMT INQ TASK for tasks that are waiting on interval control.
Next, 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.

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 10.

Notes:
1. Trace point AP_F322 is used to report that system task APTIX has been resumed. APTIX has the job of “waking up” your task on expiration of the specified interval.
   The task number for APTIX is, in this case, X'00006', and this value is shown on the trace entry.
2. Trace point DS 0004 is on entry to the dispatcher SUSPEND/RESUME interface. This function is stated explicitly in the header.
   TASK-00006 indicates that the trace entry is for system task APTIX.
   SUSPEND_TOKEN(01040034) shows that APTIX is requesting dispatcher domain to resume the task that was suspended for the specified interval. You will recall that a suspend token of X'01040034' was given to your task when it was first suspended.
3. Trace point DS 0005 is on exit from the dispatcher SUSPEND/RESUME interface.
   The trace entry shows RESPONSE(OK), indicating that the task whose suspend token was X'01040034' has successfully been resumed. However, note that this does not necessarily mean that the task has started to run—it has only been made “dispatchable”. For example, it still needs to wait for a TCB to become available.

Now look forward in the trace, and locate a trace entry made from trace point AP 00F3 and showing your task number. This and the next entry conclude the DELAY request for your task. They are shown in Figure 11 on page 80.
Notes:

1. Trace point AP 00F3 is on EXIT from interval control program. Field REQ(0005) shows that this is so, and it also shows that the response was normal. Anything other than a normal response would result in a value other than X’00’ for the first byte of the REQ field.

2. Trace point AP 00E1 is on EXIT from the EXEC interface program. This is shown by bits 0–3 of the second byte of the REQ value, X’F4’.

   The values shown for FIELD A and FIELD B show that no exception condition was detected.

That is the end of the DELAY processing, and the task that was suspended should have been resumed.

When you look at your own trace table, 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 Chapter 11, “Dealing with storage violations,” on page 193. 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.

Table 6 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 6. Resource types for file control waits

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
<th>RLS or non-RLS access mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFDTWAIT</td>
<td>The task is waiting for a request to the CFDT server to complete.</td>
<td>N/A. The wait is caused by access to a coupling facility data table.</td>
</tr>
<tr>
<td>CFDTPOOL</td>
<td>The task is waiting for a CFDT “maximum requests” slot to become available.</td>
<td>N/A. The wait is caused by access to a coupling facility data table.</td>
</tr>
<tr>
<td>CFDTPOOL</td>
<td>The task is waiting for a CFDT “locking request” slot to become available.</td>
<td>N/A. The wait is caused by access to a coupling facility data table.</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>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 91.</td>
<td>Non-RLS</td>
</tr>
</tbody>
</table>
Table 6. Resource types for file control waits (continued)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
<th>RLS or non-RLS access mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCACWAIT</td>
<td>CICS is waiting for the last RLS file to close after an SMSVSAM failure.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCBFWAIT</td>
<td>The task is waiting for a VSAM buffer.</td>
<td>Non-RLS</td>
</tr>
<tr>
<td>FCCAWAIT</td>
<td>CICS is waiting on a VSAM control ACB request.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCCFQR</td>
<td>CICS is waiting for the SMSVSAM server to notify CICS of a new quiesce request</td>
<td>RLS</td>
</tr>
<tr>
<td>FCCFQS</td>
<td>CICS is waiting for a user task to issue a new quiesce request.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCCRWAIT</td>
<td>CICS is waiting for last RLS control ACB request to complete during cleanup after SMSVSAM failure.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCDWWAIT</td>
<td>The task is waiting for VSAM to complete update processing.</td>
<td>Non-RLS</td>
</tr>
<tr>
<td>FCFSWAIT</td>
<td>The task is waiting to change the state of a file.</td>
<td>Both</td>
</tr>
<tr>
<td>FCIOWAIT</td>
<td>The task is waiting for I/O on a disk volume.</td>
<td>Non-RLS</td>
</tr>
<tr>
<td>FCIRWAIT</td>
<td>The task is waiting for the recoverable file control environment to be rebuilt.</td>
<td>Both</td>
</tr>
<tr>
<td>FCPSWAIT</td>
<td>The task is waiting for a private string.</td>
<td>Both</td>
</tr>
<tr>
<td>FCQUIES</td>
<td>The task is waiting for a quiesce request to complete.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCRRAWAIT</td>
<td>The task is waiting for file control to process non-recoverable requests at CICS restart.</td>
<td>Both</td>
</tr>
<tr>
<td>FCRBWAIT</td>
<td>The task is waiting for file control to process recoverable requests at CICS restart.</td>
<td>Both</td>
</tr>
<tr>
<td>FCRDWAIT</td>
<td>The task is waiting for a drain of the RLS control ACB to complete.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCRPWAIT</td>
<td>The task is waiting for file control initialization to complete.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCRRRWAIT</td>
<td>The task is waiting for a dynamic RLS restart to complete.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCRVWAIT</td>
<td>The task is waiting for I/O on a disk volume.</td>
<td>RLS</td>
</tr>
<tr>
<td>FCSHUTIM</td>
<td>An immediate shutdown has been invoked. Any task attempting to invoke VSAM will be put into a permanent wait.</td>
<td>Both</td>
</tr>
<tr>
<td>FCSRSUSP</td>
<td>The task is waiting for a shared resource string.</td>
<td>Non-RLS</td>
</tr>
</tbody>
</table>
Table 6. Resource types for file control waits (continued)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
<th>RLS or non-RLS access mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCTISUSP</td>
<td>The task is waiting for a VSAM transaction ID.</td>
<td>Non-RLS</td>
</tr>
<tr>
<td>FCXWAIT</td>
<td>The task is waiting for exclusive control of a VSAM control interval.</td>
<td>Non-RLS</td>
</tr>
</tbody>
</table>

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 the maxreqs 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 & FCCRWAIT—wait for SMSVSAM cleanup

The only task that can wait on resource types FCACWAIT and FCCRWAIT is CSFR, the task that performs cleanup after failure of the SMSVSAM server. This is the server that CICS file control uses for any VSAM request it issues in RLS mode.

Cleanup 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 cleanup 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 cleanup to complete, it is waiting on resource type FCCRWAIT.

FCACWAIT and FCCRWAIT are RLS-related waits only.

Resource type FCBFWAIT—waits for VSAM buffers

If your task is waiting on resource type FCBFWAIT, it means that a VSAM buffer is not currently available. You can specify the number of VSAM data buffers and VSAM index buffers in the 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 FCDWWAIT—wait for VSAM to complete update processing

If your task is waiting on resource type FCDWWAIT, it has received a VSAM response which might indicate that your task is trying to read a record. 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 using either 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 a 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 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—for example, to CLOSE or DISABLE it—but another task is, or other tasks are, 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 130.
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 type FCXCWAIT—VSAM exclusive control wait” on page 88.

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 FCPSWAIT and FCSRSUSP—waits for VSAM strings**

If your task is waiting on either of resource types FCPSWAIT or FCSRSUSP, it means that it cannot get a VSAM string. FCPSWAIT shows that the wait is for a private string, and FCSRSUSP shows that the wait is for a shared resource string. You can purge the task from the system, if the task is purgeable.

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.

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 re-establish 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 7 for advice if you suspect this.
- 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 type FCXCWAIT—VSAM exclusive control wait**

If your task is waiting on resource type FCXCWAIT, it means that it cannot get exclusive control of a VSAM control interval at the present time. Another task already has shared or exclusive control of the control interval, so your task is suspended pending the release of that control interval. An exclusive control wait on resource type FCXCWAIT occurs within CICS, unlike the similar wait on FCIOWAIT, which occurs within VSAM. See page "Resource type FCIOWAIT—wait for VSAM I/O (non-RLS)” on page 85.
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.

Waiting on this resource can occur only for files accessed in non-RLS mode.

The possibility that a task is deadlocked, waiting on itself or another task for release of the control interval, is dealt with in the next section.

**Exclusive control deadlock**

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 FCXWAIT and causing a task to wait (either itself or another task), causing a deadlock, the task is abended either with abend code AFCF or AFCG at the time that it makes the request for exclusive control.

A task that is abended with abend code AFCF would have been waiting for exclusive control of a VSAM control interval of which another task has shared or exclusive control.

A task that is abended with abend code AFCG would have been waiting for exclusive control of a VSAM control interval of which it has shared control.

See *CICS Messages and 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 abended task, and attempt to find the one in error. It is likely to be one that provides successive browse and update facilities. When you have found the programs associated with the task, turn to "How tasks can become deadlocked waiting for exclusive control" for guidance about finding how the error might have occurred.

**How tasks can become deadlocked waiting for exclusive control**

Tasks can become deadlocked waiting for exclusive control of a CI only when they have shared control of the CI and then attempt to get exclusive control without relinquishing shared control first. This can only occur for VSAM shared resource data sets accessed in non-RLS mode.

For the deadlock to occur, a transaction must first issue a VSAM READ SEQUENTIAL request via EXEC CICS STARTBR. This is a VSAM “shared control” operation. It must then issue some VSAM request requiring exclusive control of the CI without first ending the shared control operation.

The requests that require exclusive control of the CI are:

- VSAM READ UPDATE, via EXEC CICS READ UPDATE and, subsequently, EXEC CICS REWRITE.
Exclusive control of the CI is not acquired until after the initial read is complete, but it happens automatically after that and the CI is not released until the record has been rewritten.

- VSAM WRITE DIRECT, via EXEC CICS WRITE.
- VSAM WRITE SEQUENTIAL, via EXEC CICS WRITE MASSINSERT.

VSAM handles requests requiring exclusive control on a data set that is already being used in shared control mode by queueing them internally. VSAM returns control to CICS, but transactions waiting for exclusive control remain suspended.

**Example of code causing an exclusive control deadlock**

The following sequence of EXEC commands would cause an exclusive control deadlock to occur.

The first command causes shared control to be acquired:

```
EXEC CICS STARTBR
FILE(myfile)
RIDFLD(rid-area)
```

This causes no problems. The next command at first acquires shared control while the record is read into “input-area”. When an attempt is subsequently made to get exclusive control, deadlock occurs because the task that wants exclusive control is also the task that is preventing it from being acquired.

```
EXEC CICS READ
FILE(myfile)
INTO(input-area)
RIDFLD(rid-area)
UPDATE
```

The following sequence of commands would not cause deadlock to occur, because the transaction relinquishes its shared control of the CI by ending the browse before attempting to get exclusive control of it.

The first command causes shared control to be acquired:

```
EXEC CICS STARTBR
FILE(myfile)
RIDFLD(rid-area)
```

The next command causes shared control to be relinquished:

```
EXEC CICS ENDBR
FILE(myfile)
```

The next command initially causes shared control to be acquired. The record is read into “input-area”, and then exclusive control is acquired in place of shared control.

```
EXEC CICS READ
FILE(myfile)
INTO(input-area)
RIDFLD(rid-area)
UPDATE
```

The transaction now resumes. Exclusive control is relinquished following the next REWRITE or UNLOCK command on file “myfile”.

---

90 CICS TS for z/OS: CICS Problem Determination Guide
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 7 shows the type of lock that each of the “FC” resource names represents.

Table 7. Resource/pool names and lock types

<table>
<thead>
<tr>
<th>Resource or pool name</th>
<th>Lock type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCDSRECD</td>
<td>VSAM or CICS-maintained data table record</td>
</tr>
<tr>
<td>FCFLRECD</td>
<td>BDAM or user-maintained data table record</td>
</tr>
<tr>
<td>FCDSRNGE</td>
<td>KSDS key range</td>
</tr>
<tr>
<td>FCDSLDMN</td>
<td>VSAM load mode</td>
</tr>
<tr>
<td>FCDSESWR</td>
<td>ESDS write</td>
</tr>
<tr>
<td>FCFLUMTL</td>
<td>User-maintained data table load</td>
</tr>
</tbody>
</table>

For general information about enqueue waits, see “Investigating enqueue waits” on page 72.

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 Chapter 10, “Dealing with incorrect output,” on page 169.
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.

Note that the loader does not suspend a task while a program is loaded if it is the first one to ask for that program.

If the requested program is not loaded quickly, the reasons for the wait need to be investigated. The possible reasons for the wait, and the ways you should investigate them are:

1. The system could be short on storage (SOS), so only system tasks can be dispatched.
   - To check if the system is short on storage use the CEMT transaction, submitting CEMT I SYS SOSSTATUS. 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 the CICS Performance Guide.

2. There could be an I/O error on a library.
   - Check for messages that might indicate this. If you find one, investigate the reason why the I/O error occurred.

3. There could be an error within 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

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

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

Collecting the evidence

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

Turn to the lock manager summary information Figure 12 on page 94 shows an example of this).
Table 8 on page 95 describes each of the fields in the lock manager summary information.

<table>
<thead>
<tr>
<th>LOCK NAME</th>
<th>LOCK ADDRESS</th>
<th>OWNER ADDRESS</th>
<th>MODE</th>
<th>COUNT</th>
<th># LOCKS</th>
<th># LOCKS -&gt; QUEUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMLOCK</td>
<td>03B05100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DSITLOCK</td>
<td>03B05200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LD_GBLOK3</td>
<td>03B05230</td>
<td>03B0A000</td>
<td>EXCL</td>
<td>1</td>
<td>1</td>
<td>03B09700</td>
</tr>
<tr>
<td>LD_GBLOK2</td>
<td>03B05260</td>
<td>03B0AB00</td>
<td>EXCL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DMPCKNM</td>
<td>03B05290</td>
<td>03B0AB00</td>
<td>EXCL</td>
<td>35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CCSERLCK</td>
<td>03B052C0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 12. Lock manager summary information**

<table>
<thead>
<tr>
<th>LOCK NAME</th>
<th>ADDRESS</th>
<th>NEXT ADDRESS</th>
<th>OWNER ADDRESS</th>
<th>MODE</th>
<th>SUSPEND ADDRESS</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD_GBLOK3</td>
<td>03B09700</td>
<td>00000000</td>
<td>03B0A300</td>
<td>EXCL</td>
<td>01000000</td>
<td></td>
</tr>
</tbody>
</table>
Table 8. Fields in the lock manager summary information

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK NAME</td>
<td>The name given to the lock by the domain that originally issued the ADD_LOCK command.</td>
</tr>
<tr>
<td>LOCK TOKEN</td>
<td>The token assigned by the lock manager to uniquely identify the lock.</td>
</tr>
<tr>
<td>OWNER</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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.                                                                |
| MODE        | The lock mode. It can be:  
| EXCL        | The lock is exclusive—only one task can hold the lock at any one time. The lock requester is identified in the OWNER field.                     |
| SHR         | The lock is shared—several tasks can hold the lock.                                                                                          |
| SUSPEND TOKEN | The dispatcher suspend token for the suspended task.                                                                                      |
| STATUS      | The status of the suspended task. It can be:  
| Blank       | The task is waiting to acquire the lock.                                                                                                      |
| DELETED     | The suspended task has been deleted from the queue. This occurs only if the lock is deleted.                                                 |
| PURGED      | The task was purged while waiting to acquire the lock.                                                                                       |
The first step is to establish which lock the suspended task is waiting on. Obtain the KE_TAS number from the dispatcher domain summary for the suspended task and match this with an OWNER in the ‘LOCK WAIT QUEUE’ section of the lock manager summary information.

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

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

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

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

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

**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**

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

A task may fail to run if the system is at MXT, or if the task is defined in a transaction class that is at its MAXACTIVE 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 (Figure 13 on page 99) 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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tran id</td>
<td>The primary transaction id associated with the transaction</td>
</tr>
<tr>
<td>Tran num</td>
<td>The unique transaction number assigned to the transaction</td>
</tr>
<tr>
<td>Txn Addr</td>
<td>The address of the transaction control block</td>
</tr>
<tr>
<td>TxdAddr</td>
<td>The address of the transaction definition instance associated with the transaction</td>
</tr>
</tbody>
</table>
**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.

---

**Figure 13** shows a transaction summary.

<table>
<thead>
<tr>
<th>Tran</th>
<th>Tran</th>
<th>TxnAddr</th>
<th>Start</th>
<th>Sys</th>
<th>Status</th>
<th>DS</th>
<th>Facility</th>
<th>Facility</th>
<th>AP</th>
<th>PG</th>
<th>XS</th>
<th>US</th>
<th>RM</th>
<th>SM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSTP</td>
<td>00003</td>
<td>10106200</td>
<td>C</td>
<td>Yes</td>
<td>ACT</td>
<td>n/a</td>
<td></td>
<td>10164600</td>
<td>01000000</td>
<td>01017800</td>
<td>00000000</td>
<td>00000000</td>
<td>1016C00</td>
<td>10089020</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSNE</td>
<td>0031</td>
<td>10106100</td>
<td>C</td>
<td>Yes</td>
<td>ACT</td>
<td>n/a</td>
<td></td>
<td>10164C00</td>
<td>01000000</td>
<td>01017800</td>
<td>00000000</td>
<td>00000000</td>
<td>1016C058</td>
<td>11542054</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC06</td>
<td>10056</td>
<td>10E2B200</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>089601C7</td>
<td>Terminal</td>
<td>10E167A0</td>
<td>1124F600</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C9A0</td>
<td>11543610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC12</td>
<td>10058</td>
<td>10E34C00</td>
<td>SD</td>
<td>No</td>
<td>ACT</td>
<td>050601AD</td>
<td>Terminal</td>
<td>001DE600</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C9F8</td>
<td>11545114</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA03</td>
<td>93920</td>
<td>10AFF200</td>
<td>T</td>
<td>No</td>
<td>TCL</td>
<td>DFHTCL03</td>
<td></td>
<td>11214BD0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA03</td>
<td>93960</td>
<td>10E2D200</td>
<td>T</td>
<td>No</td>
<td>TCL</td>
<td>DFHTCL03</td>
<td></td>
<td>10E573F0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA03</td>
<td>93967</td>
<td>10AFF200</td>
<td>T</td>
<td>No</td>
<td>TCL</td>
<td>DFHTCL03</td>
<td></td>
<td>10E2C3F0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA03</td>
<td>94001</td>
<td>10E34800</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>DF(AKCC)</td>
<td></td>
<td>10E2C3F0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA02</td>
<td>95140</td>
<td>10E3D000</td>
<td>T</td>
<td>No</td>
<td>Act</td>
<td>DFHTCL03</td>
<td></td>
<td>10E2C5E8</td>
<td>00057000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C790</td>
<td>11544754</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA02</td>
<td>95175</td>
<td>10E3D000</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>DFHTCL03</td>
<td></td>
<td>10E937E0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C7E8</td>
<td>115457C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA02</td>
<td>95205</td>
<td>10E2D600</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>DF(AKCC)</td>
<td></td>
<td>10E837E0</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C7E8</td>
<td>115457C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA04</td>
<td>96637</td>
<td>10E2D800</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>DFHTCL03</td>
<td></td>
<td>10E05BD0</td>
<td>00057600</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C7E8</td>
<td>115457C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA04</td>
<td>96649</td>
<td>10E3D000</td>
<td>T</td>
<td>No</td>
<td>ACT</td>
<td>DFHTCL03</td>
<td></td>
<td>10E05BD0</td>
<td>00057600</td>
<td>00000000</td>
<td>00000000</td>
<td>10114023</td>
<td>1016C7E8</td>
<td>115457C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 13. Transaction summary**

**Notes for Figure 13**

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

MXT summary

The MXT summary indicates whether CICS is currently at MXT and shows the current number of queued and active transactions. To check the status of an individual transaction, consult the main transaction summary (Figure 13 on page 99).

==XM: MXT SUMMARY

| Maximum user tasks (MXT): | 7 |
| System currently at MXT:    | Yes |
| Current active user tasks:  | 7 |
| Current queued user tasks:  | 2 |
| Peak active user tasks:     | 7 |
| Peak queued user tasks:     | 2 |
| Times at MXT limit:         | 1 |

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

Transaction class summary

The transaction class summary lists each transaction class that is currently installed. For each class, the current number of active and queued transactions is shown. A transaction class is at its MAXACTIVE limit if its ‘current active’ total is greater than or equal to its ‘max active’ setting. If a transaction class is at its MAXACTIVE limit, a number of transactions may be queueing in that transaction class. The transaction id and number of each queued transaction is listed with its transaction class (for example, transaction classes DFHCTL01, DFHCTL02, and DFHCTL03 in Figure 14).

==XM: TCLASS SUMMARY

<table>
<thead>
<tr>
<th>Tclass Name</th>
<th>Max Active</th>
<th>Purge Thresh</th>
<th>Current Active</th>
<th>Current Queued</th>
<th>Total Attaches</th>
<th>Queuing TranNum</th>
<th>Queuing TranId</th>
<th>Queuing Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTCL01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95175</td>
<td>TA02</td>
<td>18:00:19.677</td>
</tr>
<tr>
<td>DFHTCL02</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>95175</td>
<td>TA02</td>
<td>18:00:24.624</td>
</tr>
<tr>
<td>DFHTCL03</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>29</td>
<td>93929</td>
<td>TA03</td>
<td>17:55:40.584</td>
</tr>
<tr>
<td>DFHTCL04</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>93649</td>
<td>TA04</td>
<td>18:06:04.348</td>
</tr>
<tr>
<td>DFHTCL05</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93968</td>
<td>TA03</td>
<td>17:55:42.230</td>
</tr>
<tr>
<td>DFHTCL06</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93967</td>
<td>TA03</td>
<td>17:55:52.253</td>
</tr>
<tr>
<td>DFHTCL07</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93968</td>
<td>TA03</td>
<td>17:55:52.253</td>
</tr>
<tr>
<td>DFHTCL08</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93967</td>
<td>TA03</td>
<td>17:55:52.253</td>
</tr>
<tr>
<td>DFHTCL09</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93968</td>
<td>TA03</td>
<td>17:55:52.253</td>
</tr>
<tr>
<td>DFHTCL10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93967</td>
<td>TA03</td>
<td>17:55:52.253</td>
</tr>
</tbody>
</table>

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

Figure 14. 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 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.

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.

You can use CEMT online to identify deadlocked transactions, and identify the resources they hold.

The CEMT INQUIRE UOWENQ 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. 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.

### An example of deadlock diagnosis using CEMT INQUIRE UOWENQ

Consider the following example. The user of task 32 complains that his terminal is locked: he cannot enter data and his terminal does not respond to any prompt.

If you enter CEMT INQUIRE TASK at a different terminal at this time, a display similar to the following might appear:

<table>
<thead>
<tr>
<th>INQUIRE TASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS: RESULTS - OVERTYPE TO MODIFY</td>
</tr>
<tr>
<td>Tas(0000025) Tra(CEMT) Fac(T773) Run Ter Pri( 255 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545D8C01)</td>
</tr>
<tr>
<td>Tas(0000028) Tra(TDUP) Fac(T774) Sus Ter Pri( 001 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545CAD227) Hty(ZCIOWAIT) Hva(DFHZARQ1)</td>
</tr>
<tr>
<td>Tas(0000032) Tra(FUPD) Fac(T775) Sus Ter Pri( 001 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545D004) Hty(ENQUEUE) Hva(FCDSRECD)</td>
</tr>
<tr>
<td>Tas(0000035) Tra(FUPD) Fac(T784) Sus Ter Pri( 001 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545BC357) Hty(ENQUEUE) Hva(FCDSRECD)</td>
</tr>
<tr>
<td>Tas(0000039) Tra(FUPD) Fac(T778) Sus Ter Pri( 001 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545F92F403) Hty(ENQUEUE) Hva(FCDSRECD)</td>
</tr>
<tr>
<td>Tas(0000042) Tra(FUP2) Fac(T783) Sus Ter Pri( 001 )</td>
</tr>
<tr>
<td>Sta(TO) Use(CICSUSER) Uow(AA8E950545C1B9A) Hty(ENQUEUE) Hva(FCDSRECD)</td>
</tr>
</tbody>
</table>
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). But, at this stage, you cannot tell which (if any) task has control of this resource.

If you now enter CEMT INQUIRE UOWENQ at the same terminal, you can see the enqueues held in the system - you can see which tasks own which resources and which tasks are waiting for those resources.

A screen similar to the following might be displayed:

<table>
<thead>
<tr>
<th>INQUIRE UOWENQ TASK(32)</th>
<th>STATUS: RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uow(AABE950545BC04C01)</td>
<td>Tra(CMNT) Tas(0000025) Act Exe Own</td>
</tr>
<tr>
<td>Uow(AABE950545Clock27)</td>
<td>Tra(TDUP) Tas(0000028) Act Tdq Own</td>
</tr>
<tr>
<td>Uow(AABE950545C004C04)</td>
<td>Tra(FUPD) Tas(0000032) Act Dat Own</td>
</tr>
<tr>
<td>Uow(AABE950545D000C04)</td>
<td>Tra(FUPD) Tas(0000035) Act Dat Wai</td>
</tr>
<tr>
<td>Uow(AABE97F9952F403)</td>
<td>Tra(FUP2) Tas(0000039) Act Dat Wai</td>
</tr>
<tr>
<td>Uow(AABE950545D000C04)</td>
<td>Tra(TSUP) Tas(0000034) Ret Tsq Own</td>
</tr>
<tr>
<td>Uow(AABE97F9952F403)</td>
<td>Tra(FUP2) Tas(0000039) Act Dat Own</td>
</tr>
<tr>
<td>Uow(AABE950545D000C04)</td>
<td>Tra(FUPD) Tas(0000032) Act Dat Wai</td>
</tr>
<tr>
<td>Uow(AABE97F9952F403)</td>
<td>Tra(FUPD) Tas(0000042) Act Dat Own</td>
</tr>
</tbody>
</table>

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 which owns the enqueue. Owners and waiters on other regions are not displayed.

You can clarify this display in a busy system 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 INQUIRE UOWENQ command: CEMT INQUIRE UOWENQ TASK(32).

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).

In order 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(AABE950545DAC004)
  Transid(FUPD)
  Taskid(0000032)
  State(Active)
  Type(Dataset)
  Relation(Owner)
  Resource(ACCT.CICS640.ACCTFILE)
  Qualifier(SMITH)
  Netuowid(.GBIBMIYA.IYA2T774.n......)
  Enqfails(00000000)

This shows you details of the enqueue that task 32 owns.
Expanding the second entry shows the enqueue that task 32 is waiting for:

```
INQUIRE UOWENQ TASK(32)
RESULT
  Uowenq
    Uow(AA8E9505450AC004)
    Transid(FUPD)
    Taskid(00000032)
    State(Active)
    Type(Dataset)
    Relation(Waiter)
    Resource(INDX.CICS640.ACIXFILE)
    Qualifier(SMITH)
    Netuowid(...GB1BMYA.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.CICS640.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.CICS640.ACIXFILE data set.

So now you need to 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. Enter CEMT INQUIRE UOWENQ RESOURCE(INDX.CICS640.ACIXFILE) QUALIFIER(SMITH) This shows the task that owns the enqueue that is being waited on.

```
INQUIRE UOWENQ RESOURCE(INDX.CICS640.ACIXFILE) QUALIFIER(SMITH)
STATUS: RESULTS
  Uow(AA8E97FE9592F403) Tra(FUP2) Tas(00000039) Act Dat Own
  Uow(AA8E9505450AC004) Tra(FUPD) Tas(00000032) Act Dat Wai
```

This shows you that another task, task 39, owns the enqueue that task 32 is waiting on. In order to find out why task 39 is holding this enqueue, filter the CEMT command again for task 39. Enter CEMT INQUIRE UOWENQ TASK(39).

```
INQUIRE UOWENQ TASK(39)
STATUS: RESULTS
  Uow(AA8E97FE9592F403) Tra(FUP2) Tas(00000039) Act Dat Wai
  Uow(AA8E97FE9592F403) Tra(FUP2) Tas(00000039) Act Dat Own
```

This shows you that task 39 is also waiting for an enqueue. If you expand the entry which indicates the waiting state, you may see a display similar to the following:
This shows you that task 39 is waiting for the enqueue on record “SMITH” in the ACCT.CICS640.ACCTFILE data set. This is the enqueue that task 32 owns. You can now see that the deadlock is between tasks 32 and 39.

If you filter by the RESOURCE and QUALIFIER of this enqueue, you can confirm that this is the case. This also shows that task 35 also waits on the enqueue owned by task 32.

You are now in a position of knowing which transaction(s) to cancel and investigate further.

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 wish to automate deadlock detection and resolution, these commands are of great benefit.

**Resolving deadlocks in a sysplex**

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 or CEC 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. CPSM tries to assist in the determination of which task to purge to free-up the system. CPSM 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:
1. Display the suspended task’s details and determine what the suspend reason is (CPSM uses EXEC CICS INQUIRE TASK internally). Note the internal UOW id (the RM UOW).
   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 via the UOWENQ display (which CPSM is building from EXEC CICS INQUIRE UOWENQ(uow) Browse for this UOW:id). From this display you can get the enqueue name that this task is waiting for.

3. Display the details of this enqueue (CPSM uses the EXEC CICS INQUIRE UOWENQ RESOURCE(name) browse facility to determine what UOWs are interested in this Enqueue (holder and waiters).
   You are now in a position to analyze the problem to determine the cause of the problem.

### Resolving in-doubt and resynchronization failures

For examples of how to resolve in-doubt and resynchronization failures, see the [CICS Intercommunication Guide](#).

### 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:
- “CICS has stalled during initialization”
- “CICS has stalled during a run” on page 107
- “CICS has stalled during termination” on page 109

If XRF takeover by an alternate CICS system fails to complete satisfactorily, that might also appear to you as a CICS stall. For more information, see the [CICS/ESA 4.1 Problem Determination Guide](#).

### 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 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 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 the CICS Customization Guide.

CICS has stalled during a run

If a CICS region that has been running normally stalls, so that it produces no output and accepts no input, the scope of the problem is potentially system-wide. The problem might be confined exclusively to CICS, or it could be caused by any other task running under 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 Chapter 8, “Dealing with loops,” on page 143 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 paragraphs that follow. 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?

It is possible that the system definition parameters for your system are causing it to stall, possibly at a critical loading. Take a look at what has been specified, paying particular attention to these items:

- The CICS maximum tasks (MXT) and transaction class (MAXACTIVE) limits. If these are too low, new tasks could fail to be attached. If you suspect one of these limits is the cause of the stall, read “Are MXT or transaction class limits causing the stall?” on page 108 for a way of getting further evidence.
The system region exit time. If this is set too high, CICS might relinquish control to the operating system for longer than intended when it has no work to do, perhaps giving the impression of a stall.

ICVR, the runaway task time interval. If this is set too high, a runaway task could stop other tasks from running for a relatively long time. It can have a value up to 2 700 000 milliseconds, in which case a runaway task would not time out for 45 minutes. CICS could, in the meantime, be stalled. If the ICVR is set to 0, the runaway task does not time out at all.

You should already have an indication if the ICVR is the problem, from the CPU usage (see above).

ICVTSD, the terminal scan delay interval. The effect differs for VTAM and non-VTAM terminals, but if its value is not appropriate for your system the effect could make you think that CICS is stalled.

For more details about the choice of these and other system definition parameters, see the CICS Performance Guide.

Is the system short on storage?
If storage is under stress, storage manager statistics indicate that a storage stress situation has occurred (‘Times went short on storage’). In addition, if the SOS is caused by a suspended GETMAIN or if CICS is unable to alleviate the situation by releasing programs with no current user, and slowing the attachment of new tasks:

• A message is sent to the console saying that CICS is short on storage (DFHSM0131I for storage below the 16 megabyte line, DFHSM0133I for storage above the 16 megabyte line)

• The storage manager statistic ‘Times went short on storage’ is updated.

CICS can go short on storage independently in any DSA. You may see tasks suspended on any of the resource types, CDSA, SDSA, RDSA, USDA, ECDSA, ESDSA, ERDSA, or EUDSA.

Are MXT or transaction class limits causing the stall?
Before new transactions can be attached for the first time, they must qualify under the MXT and transaction class limits. In a system that is running normally, tasks run and terminate and new transactions are attached, even though these limits are reached occasionally. It is only when tasks can neither complete nor be purged from the system that CICS can stall as a result of one of these limits being reached.

Look first at the transaction manager summary in the formatted system dump.

Investigate the tasks accepted into the MXT set of tasks to see if they are causing the problem. XM dump formatting formats the state of MXT and provides a summary of the TCLASSes and of the transactions waiting for acceptance into each TCLASS.

Now look at the 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 VTAM 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 VTAM status is OPEN, the problem could be associated with processing done in the VTAM part of your system or with processing done in the CICS part of your system. If it appears that there is a communication problem, consider using either CICS VTAM exit tracing or VTAM buffer tracing. For guidance about using these techniques, see Chapter 15, “Using traces in problem determination,” on page 225.
- If the VTAM status is CLOSED, CICS cannot use VTAM 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 129.

**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® master terminal and display the active sessions. If necessary, close down the network using the VARY NET,INACT,ID=applid command. This should enable CICS to resume its termination sequence. If it does not, you might need to cancel the CICS job. If this does happen, consider whether any PLT program running in the second quiesce stage could be in error. If such a program did not follow the strict protocols that are required, it could cause CICS to stall during termination. For programming information about PLT programs, see the CICS Customization Guide.
Chapter 7. 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 the CICS Diagnosis Reference. 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 52.

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 the CICS Customization Guide.

If you want guidance about using online or offline techniques to investigate waits, see "Techniques for investigating waits" on page 51.

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 9 on page 113. 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 concept information on the following waits:
- "The resources on which tasks in a CICS system can wait" on page 113
- "Dispatcher waits" on page 125
- "CICS DB2 waits" on page 126
- "DBCTL waits" on page 128
- "Investigating storage waits" on page 67
- "EDF waits" on page 129
- "Investigating terminal waits" on page 57
- "Log manager waits" on page 129
- "Task control waits" on page 130
- "VTAM terminal control waits" on page 133
- "Interregion and intersystem communication waits" on page 134
- "IIOP waits" on page 135
- "Transient data waits" on page 135
- "XRF alternate system waits" on page 139
- "CICS system task waits" on page 140
- "PEI waits" on page 141
- "Recovery manager waits" on page 141

The resources on which tasks in a CICS system can wait

Table 9 on page 113 shows all the resources on which tasks in a CICS system can wait. 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 on some of the resources. For each such resource, there is a page reference showing you where to look for guidance about dealing with the wait. The 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 on 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 140.
### Table 9. Resources on which a suspended task might be waiting

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Purge status</th>
<th>Resource name</th>
<th>Suspending module</th>
<th>DSSR call and WLM wait type</th>
<th>Task</th>
<th>Where to look next</th>
</tr>
</thead>
<tbody>
<tr>
<td>(none)</td>
<td>(none)</td>
<td>DFHDUIO</td>
<td>WAIT_MVS IO</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>(none)</td>
<td>DFHRMSL7</td>
<td>WAIT_MVS TIMER</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>(none)</td>
<td>DFHZNAC</td>
<td>SUSPEND See note on page 123</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>DLCNTRL</td>
<td>DFHDBCCT</td>
<td>WAIT_MVS See note on page 123</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>DLCONECT</td>
<td>DFHDBCQN</td>
<td>WAIT_MVS OTHER_PRODUCT</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>DMWTQUEU</td>
<td>DFHDMWQ</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>No, No</td>
<td>LQUEUE</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating lock manager waits&quot; on page 93</td>
<td></td>
</tr>
<tr>
<td>ADAPTER</td>
<td>No, No</td>
<td>FEPI_RQE</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>See note 2 on page 123</td>
<td></td>
</tr>
<tr>
<td>ALLOCATE</td>
<td>Yes, Yes</td>
<td>TCTETETI value</td>
<td>SUSPEND See note on page 124</td>
<td>User</td>
<td>&quot;Interregion and intersystem communication waits&quot; on page 134</td>
<td></td>
</tr>
<tr>
<td>ALP_TERM</td>
<td>(none)</td>
<td>DFHALRC</td>
<td>WAIT_OLD MISC</td>
<td>System only</td>
<td>&quot;Recovery manager waits&quot; on page 141</td>
<td></td>
</tr>
<tr>
<td>Any_MBCB</td>
<td>No, No</td>
<td>transient data queue name</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;Transient data waits&quot; on page 135</td>
<td></td>
</tr>
<tr>
<td>Any_MRCB</td>
<td>No, No</td>
<td>transient data queue name</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;Transient data waits&quot; on page 135</td>
<td></td>
</tr>
<tr>
<td>AP_INIT</td>
<td>ECBTCP</td>
<td>DFHAPSIP</td>
<td>WAIT_OLD MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>AP_INIT</td>
<td>SIPDMTEC</td>
<td>DFHAPSIP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>AP_INIT</td>
<td>TCTVCECB</td>
<td>DFHSII1</td>
<td>WAIT_OLD MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>AP_INIT</td>
<td>ZGRPECB</td>
<td>DFHSII1</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>AP_QUIES</td>
<td>CSASSI2</td>
<td>DFHSTP</td>
<td>WAIT_OLD MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>AP_QUIES</td>
<td>SHUTECB</td>
<td>DFHSTP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>APRDR</td>
<td>INITIAL</td>
<td>DFHAPRDR</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;Recovery manager waits&quot; on page 141</td>
<td></td>
</tr>
<tr>
<td>APRDR</td>
<td>RECOVER</td>
<td>DFHAPRC</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;Recovery manager waits&quot; on page 141</td>
<td></td>
</tr>
<tr>
<td>AP_TERM</td>
<td>STP_DONE</td>
<td>DFHAPDM</td>
<td>WAIT_MVS LOCK</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>CCSTWAIT</td>
<td>VSMSTRNG</td>
<td>DFHCCCC</td>
<td>WAIT_OLD IO</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>CCVSAMWT</td>
<td>ASYNRESP</td>
<td>DFHCCCC</td>
<td>WAIT_MVS IO</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>CCVSAMWT</td>
<td>EXCLOGER</td>
<td>DFHCCCC</td>
<td>WAIT_MVS IO</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>CDB2CONN</td>
<td>No, No</td>
<td>(none)</td>
<td>DFHD2EX1</td>
<td>WAIT_MVS OTHER_PRODUCT</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
</tr>
<tr>
<td>CDB2RDYQ</td>
<td>No, No</td>
<td>name of DB2ENTRY or pool</td>
<td>DFHD2EX1</td>
<td>WAIT_MVS OTHER_PRODUCT</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
</tr>
<tr>
<td>CDB2TCB</td>
<td>No, No</td>
<td>(none)</td>
<td>DFHD2EX1</td>
<td>WAIT_MVS OTHER_PRODUCT</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
</tr>
<tr>
<td>CDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMSQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>CFDTWAIT</td>
<td>file name</td>
<td>DFHFCDO</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHFCDR</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHFCDU</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td>CFDTPPOOL</td>
<td>CFDT pool name</td>
<td>DFHFCDO</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHFCDR</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHFCDU</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td>CSNC</td>
<td>MROQUEUE</td>
<td>DFHCRNP</td>
<td>WAIT_MVS</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 7 on page 123</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td>No, No</td>
<td>LOT_ECB</td>
<td>DFHD2EX1</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
<td></td>
</tr>
<tr>
<td>DB2_INIT</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHD2IN1</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
<td></td>
</tr>
<tr>
<td>DB2CDISC</td>
<td>Yes, Yes</td>
<td>name of DB2CONN</td>
<td>DFHD2TM</td>
<td>WAIT OLDC MISC</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
</tr>
<tr>
<td>DB2EDISA</td>
<td>Yes, Yes</td>
<td>name of DB2ENTRY</td>
<td>DFHD2TM</td>
<td>WAIT OLDC MISC</td>
<td>User</td>
<td>&quot;CICS DB2 waits&quot; on page 126</td>
</tr>
<tr>
<td>DBDXEOT</td>
<td>(none)</td>
<td>DFHDXSTM</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>DBDXINT</td>
<td>(none)</td>
<td>DFHXSTM</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>DBCTL</td>
<td>No, No</td>
<td>DLSUSPND</td>
<td>DFHDBSPX</td>
<td>User</td>
<td>&quot;DBCTL waits&quot; on page 128</td>
<td></td>
</tr>
<tr>
<td>DFHAIIN</td>
<td>AITM</td>
<td>DFHAIIN1</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>DFHCPIN</td>
<td></td>
<td>CPI</td>
<td>DFHCPIN1</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td>DFHPRIN</td>
<td></td>
<td>PRM</td>
<td>DFHPRIN1</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td>DFHPTTW</td>
<td>Yes</td>
<td>DFHPTTW</td>
<td>DFHPTTW</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>See [13 on page 124]</td>
</tr>
<tr>
<td>DFHSIPLT</td>
<td>EARLYPLT</td>
<td>DFHSII1</td>
<td>WAIT_MVS</td>
<td>MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td></td>
<td>LATE_PLT</td>
<td>DFHSIJ1</td>
<td>WAIT_MVS</td>
<td>MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td>DISPATCH</td>
<td>Yes, Yes</td>
<td>JVM_POOL</td>
<td>DFHDSDS4</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Dispatcher waits&quot; on page 125</td>
</tr>
<tr>
<td>DISPATCH</td>
<td>Yes, Yes</td>
<td>OPENPOOL</td>
<td>DFHDSDS4</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Dispatcher waits&quot; on page 125</td>
</tr>
<tr>
<td></td>
<td>OPEN_DEL</td>
<td>DFHDSDS4</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Dispatcher waits&quot; on page 125</td>
<td></td>
</tr>
<tr>
<td>DISPATCH</td>
<td>Yes, Yes</td>
<td>SOSMVS</td>
<td>DFHDSDS4</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Dispatcher waits&quot; on page 125</td>
</tr>
<tr>
<td>DMATTACH</td>
<td>QUIESCE</td>
<td>DFHDMDM</td>
<td>WAIT_MVS</td>
<td>MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td>ECDSA</td>
<td>(none)</td>
<td>DFHSMSQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
<td></td>
</tr>
<tr>
<td>EDF</td>
<td>Yes, No</td>
<td>DEBUGUSER</td>
<td>DFHEDFX</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;EDF waits&quot; on page 129</td>
</tr>
<tr>
<td>EJ.ST.DJ</td>
<td>Yes, Yes</td>
<td>First 16 bytes of DJAR name</td>
<td>DFHEJDG</td>
<td>WAIT_MVS</td>
<td>TIMER</td>
<td>User</td>
</tr>
<tr>
<td>EJ.ST.DC</td>
<td>Yes, Yes</td>
<td>Corbaserver name</td>
<td>DFHEJDG</td>
<td>WAIT_MVS</td>
<td>TIMER</td>
<td>User</td>
</tr>
<tr>
<td>EKWAIT</td>
<td>No, Yes</td>
<td>ATCHMSUB</td>
<td>DFHD2STR</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;CICS DB2 Guide&quot;</td>
</tr>
<tr>
<td>EKWAIT</td>
<td>No, Yes</td>
<td>CEX2TERM</td>
<td>DFHD2STP</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;CICS DB2 Guide&quot;</td>
</tr>
<tr>
<td>EKWAIT</td>
<td>No, Yes</td>
<td>DTCHMSUB</td>
<td>DFHD2STR</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;CICS DB2 Guide&quot;</td>
</tr>
<tr>
<td>EKWAIT</td>
<td>No, No</td>
<td>MSBRETRN</td>
<td>DFHD2STP</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;CICS DB2 Guide&quot;</td>
</tr>
<tr>
<td>EKWAIT</td>
<td>No, No</td>
<td>SINGLE</td>
<td>DFHEKC</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;Task control waits&quot; on page 130</td>
</tr>
<tr>
<td>ENF</td>
<td></td>
<td>NOTIFY</td>
<td>DFHDMENF</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>EXECADDR</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;EXEC CICS ENQ waits&quot; on page 74</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>EXECSTRN</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;EXEC CICS ENQ waits&quot; on page 74</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCDSESWR</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCDSDLMD</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCDSRECD</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCDSRNGE</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCFLRECD</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>FCFLUMTL</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>JOURNALS</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating enqueue waits&quot; on page 72</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>KCADDR</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>System or user</td>
<td>&quot;Investigating enqueue waits&quot; on page 72</td>
</tr>
<tr>
<td># ENQUEUE</td>
<td>Yes, Yes</td>
<td>KCSTRING</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>System or user</td>
<td>&quot;Investigating enqueue waits&quot; on page 72</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>LOGSTRMS</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating enqueue waits&quot; on page 72</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>TDNQ</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Transient data waits&quot; on page 135</td>
</tr>
<tr>
<td>ENQUEUE</td>
<td>Yes, Yes</td>
<td>TSNQ</td>
<td>DFHNQED</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits&quot; on page 69</td>
</tr>
<tr>
<td>ERDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMSEQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>ESDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMSEQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>EUDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMSEQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>FCACWAIT</td>
<td><em>CTLACB</em></td>
<td>DFHFCRD</td>
<td>WAIT_OLDC IO</td>
<td>System</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td>FCBFWAIT</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCVR</td>
<td>WAIT_OLDC IO</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>FCCAWAIT</td>
<td>No, No</td>
<td><em>CTLACB</em></td>
<td>DFHFCCA</td>
<td>WAIT_MVS OTHER_PRODUCT</td>
<td>User</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
</tr>
<tr>
<td>FCCFQR</td>
<td>(none)</td>
<td>DFHFCQR</td>
<td>WAIT_MVS See note [1] on page 123</td>
<td>System</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td>FCCFQS</td>
<td>(none)</td>
<td>DFHFCQS</td>
<td>WAIT_MVS MISC</td>
<td>System</td>
<td>&quot;Investigating file control waits&quot; on page 80</td>
<td></td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>FCCRWAIT</td>
<td>Yes, Yes</td>
<td><em>CTLACB</em></td>
<td>DFHFCRD</td>
<td>WAIT_OLDC IO</td>
<td>System</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCDWWAIT</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCVR</td>
<td>WAIT_OLDC IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCFSWAIT</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCFS</td>
<td>WAIT_OLDC IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCINWAIT</td>
<td>STATIC</td>
<td>DFHFCIN1</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>System only</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCIOWAIT</td>
<td>No, No</td>
<td>file ID</td>
<td>DFHFCBD</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCIRWAIT</td>
<td>RECOV-FC</td>
<td>DFHFCRP</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>System only</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCPSWAIT</td>
<td>Yes, Yes</td>
<td><em>CTLACB</em> file ID file ID</td>
<td>DFHFCCA DFHFCRS DFHFCVR</td>
<td>WAIT_OLDC IO WAIT_OLDC MISC</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCQUIES</td>
<td>Yes, Yes</td>
<td>fcqse_ptr (hexa-decimal)</td>
<td>DFHFCQI</td>
<td>SUSPEND See note 7 on page 123</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRAWAIT</td>
<td>Yes, Yes</td>
<td>FC_FILE</td>
<td>DFHEIFC</td>
<td>WAIT_OLDC MISC</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRBWAIT</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCFR</td>
<td>WAIT_OLDC IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRDWAIT</td>
<td>No, No</td>
<td><em>CTLACB</em></td>
<td>DFHFCRC</td>
<td>WAIT_OLDC MISC</td>
<td>System or user</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRPWAIT</td>
<td>FC-START</td>
<td>DFHFCRR</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>System only</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRRWAIT</td>
<td><em>DYRRE</em></td>
<td>DFHFCRR</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>System only</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCRVWAIT</td>
<td>No, No</td>
<td>file ID</td>
<td>DFHFCRV</td>
<td>WAIT_MVS OTHER PRODUCT</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCSRSUSP</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCVR</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
<tr>
<td>FCTISUSP</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCVR</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>Investigating file control waits on page 80</td>
</tr>
</tbody>
</table>
Table 9. Resources on which a suspended task might be waiting (continued)

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Purge status</th>
<th>Resource name</th>
<th>Suspending module</th>
<th>DSSR call and WLM wait type</th>
<th>Task</th>
<th>Where to look next</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCXWAIT</td>
<td>Yes, Yes</td>
<td>file ID</td>
<td>DFHFCVS</td>
<td>WAIT_OLDC IO</td>
<td>User</td>
<td>&quot;Investigating file control waits” on page 80</td>
</tr>
<tr>
<td>FEPRM</td>
<td>No, No</td>
<td>SZRDP</td>
<td>DFHSZRDP</td>
<td>WAIT_MVS MISC</td>
<td>CSZI</td>
<td>See note 2 on page 123</td>
</tr>
<tr>
<td>FOREVER</td>
<td>No, No</td>
<td>DFHXTMA</td>
<td>DFHXTMA</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;A user task is waiting on resource type FOREVER” on page 100</td>
</tr>
<tr>
<td>ICEXPIRY</td>
<td></td>
<td>DFHAPTIX</td>
<td>DFHAPTIX</td>
<td>SUSPEND TIMER</td>
<td>System only</td>
<td>&quot;CICS system task waits” on page 140</td>
</tr>
<tr>
<td>ICGTWAIT</td>
<td>Yes, Yes</td>
<td>terminal ID</td>
<td>DFHICP</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating interval control waits” on page 75</td>
</tr>
<tr>
<td>ICMIDNTE</td>
<td></td>
<td>DFHAPTIM</td>
<td>DFHAPTIM</td>
<td>SUSPEND TIMER</td>
<td>System only</td>
<td>&quot;CICS system task waits” on page 140</td>
</tr>
<tr>
<td>ICP_INIT</td>
<td>(none)</td>
<td>DFHICRC</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>&quot;Investigating interval control waits” on page 75</td>
<td></td>
</tr>
<tr>
<td>ICP_TERM</td>
<td>(none)</td>
<td>DFHICRC</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>&quot;Investigating interval control waits” on page 75</td>
<td></td>
</tr>
<tr>
<td>ICP_TSWT</td>
<td>(none)</td>
<td>DFHICRC</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>&quot;Investigating interval control waits” on page 75</td>
<td></td>
</tr>
<tr>
<td>ICWAIT</td>
<td>Yes, No</td>
<td>terminal ID</td>
<td>DFHICP</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating interval control waits” on page 75</td>
</tr>
<tr>
<td>IIRR</td>
<td></td>
<td>SOCBNOTI</td>
<td>DFHIIRR</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;IIOP waits” on page 135</td>
</tr>
<tr>
<td>IIRR</td>
<td></td>
<td>SOCFNOTI</td>
<td>DFHIIRR</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;IIOP waits” on page 135</td>
</tr>
<tr>
<td>IIRP</td>
<td></td>
<td>NOTI</td>
<td>DFHIIRR</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;IIOP waits” on page 135</td>
</tr>
<tr>
<td>IRLINK</td>
<td>Yes, No</td>
<td>SYSIDNT concat with session name</td>
<td>DFHZIS2</td>
<td>WAIT_MVS See note 7 on page 124</td>
<td>User</td>
<td>&quot;Investigating terminal waits” on page 57</td>
</tr>
<tr>
<td>KCCOMPAT</td>
<td>No, No</td>
<td>CICS</td>
<td>DFHXCPA</td>
<td>WAIT_OLDC LOCK</td>
<td>User</td>
<td>&quot;Task control waits” on page 130</td>
</tr>
<tr>
<td>KCCOMPAT</td>
<td>No, No</td>
<td>LIST</td>
<td>DFHXCPA</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;Task control waits” on page 130</td>
</tr>
<tr>
<td>KCCOMPAT</td>
<td>No, No</td>
<td>SINGLE</td>
<td>DFHXCPA</td>
<td>WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;Task control waits” on page 130</td>
</tr>
<tr>
<td>KCCOMPAT</td>
<td>Yes, No</td>
<td>SUSPEND</td>
<td>DFHXCPA</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Task control waits” on page 130</td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>KCCOMPAT</td>
<td>Yes, No</td>
<td>TERMINAL</td>
<td>DFHXCPA</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Task control waits&quot; on page 130 and &quot;Investigating terminal waits&quot; on page 57</td>
</tr>
<tr>
<td>LATE_PLT</td>
<td>DFHSIPLT</td>
<td>DFHSIPLT</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
<td></td>
</tr>
<tr>
<td>LG_DEFER</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2SRC</td>
<td>SUSPEND IDLE</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGDELALL</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS4</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGDELRAN</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS5</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGENDBLK</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS9</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGENDCRS</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HSJ</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LG_FORCE</td>
<td>Yes, No</td>
<td>journal name</td>
<td>DFHL2SRC</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGHARTBT</td>
<td>LG_MGRST</td>
<td>DFHLGHB</td>
<td>SUSPEND TIMER</td>
<td>System only</td>
<td>&quot;How CICS checks for the availability of the MVS logger&quot; on page 208</td>
<td></td>
</tr>
<tr>
<td>LGREDBLK</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS8</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGREDCRS</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HSG</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGSTRBLK</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS7</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGSTRCRS</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HS6</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>LGWRITE</td>
<td>No, No</td>
<td>journal name</td>
<td>DFHL2HSF</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Log manager waits&quot; on page 129</td>
</tr>
<tr>
<td>MBCB_xxx</td>
<td>No, No transient data queue name</td>
<td>DFHTDB DFHTDRM</td>
<td>SUSPEND IO</td>
<td>User</td>
<td>&quot;Transient data waits&quot; on page 135</td>
<td></td>
</tr>
<tr>
<td>MRCB_xxx</td>
<td>No, No transient data queue name</td>
<td>DFHTDB DFHTDRM</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td>&quot;Transient data waits&quot; on page 135</td>
<td></td>
</tr>
<tr>
<td>MXT</td>
<td>No, No</td>
<td>XM_HELD</td>
<td>DFHXMAT (See note on page 124)</td>
<td>User</td>
<td>&quot;Maximum task condition waits&quot; on page 96</td>
<td></td>
</tr>
<tr>
<td>PIIS</td>
<td>Yes, Yes</td>
<td>RZCBNOTI</td>
<td>DFHPIIS</td>
<td>SUSPEND MISC</td>
<td>System</td>
<td>&quot;See note on page 124&quot;</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Yes, Yes</td>
<td>program ID</td>
<td>DFHLDDMI</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating loader waits&quot; on page 93</td>
</tr>
<tr>
<td>Resource type</td>
<td>Purge status</td>
<td>Resource name</td>
<td>Suspending module</td>
<td>DSSR call and WLM wait type</td>
<td>Task</td>
<td>Where to look next</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Yes, Yes</td>
<td>program ID</td>
<td>DFHPGEX</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGIS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGLD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGLK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGLU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGPG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGRP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DFHPGXE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCP_INIT</td>
<td></td>
<td>(none)</td>
<td>DFHAPRC</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td>&quot;CICS system task waits&quot; on page 140</td>
</tr>
<tr>
<td>RDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMASQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>RMI</td>
<td>DFHERMRS</td>
<td>DFHERMRS</td>
<td>WAIT_MVS TIMER</td>
<td>System only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMCLIENT</td>
<td>No, No</td>
<td>client name</td>
<td>DFHRMCIC</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Recovery manager waits&quot; on page 141</td>
</tr>
<tr>
<td>RMUOWOB</td>
<td>No, No</td>
<td>LOGMOVE EXISTENC</td>
<td>DFHRMUO DFHRMUW DFHRMUWS DFHRMU1U DFHRMUO DFHRMUW DFHRMUWL DFHRMUWP DFHRMUWQ DFHRMU1D DFHRMU1K</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Recovery manager waits&quot; on page 141</td>
</tr>
<tr>
<td>SDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMASQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits&quot; on page 67</td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>STE</td>
<td>DFHSOCK</td>
<td>SUSPEND MISC</td>
<td>System only</td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>MAXSOCKETS</td>
<td>DFHSOEC</td>
<td>WAIT_MVS IDLE</td>
<td>System only</td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>SOCLOSE</td>
<td>DFHSOASO</td>
<td>SUSPEND MISC</td>
<td>System or user</td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>RECEIVE</td>
<td>DFHSOEC</td>
<td>WAIT_MVS IDLE</td>
<td>System or user</td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>SEND</td>
<td>DFHSOEC</td>
<td>WAIT_MVS IDLE</td>
<td>System or user</td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td>Yes</td>
<td>MISCELANCEOUS</td>
<td>DFHSOEC</td>
<td>WAIT_MVS IDLE</td>
<td>System or user</td>
<td></td>
</tr>
<tr>
<td>SODOMAIN</td>
<td>Yes</td>
<td>SO_LTEPTY</td>
<td>DFHSODM</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
</tr>
<tr>
<td>SODOMAIN</td>
<td>No</td>
<td>SO_LISTN</td>
<td>DFHSOLS</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
</tr>
<tr>
<td>SODOMAIN</td>
<td>No</td>
<td>SO_NOWORK</td>
<td>DFHSOLS</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
</tr>
</tbody>
</table>
Table 9. Resources on which a suspended task might be waiting  (continued)

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Purge status</th>
<th>Resource name</th>
<th>Suspending module</th>
<th>DSSR call and WLM wait type</th>
<th>Task</th>
<th>Where to look next</th>
</tr>
</thead>
<tbody>
<tr>
<td>SODOMAIN</td>
<td>No</td>
<td>SO_LTERDC</td>
<td>DFHSORD</td>
<td>WAIT_MVS MISC</td>
<td>System or user</td>
<td>CICS system task waits on page 140</td>
</tr>
<tr>
<td>STP_TERM</td>
<td>(none)</td>
<td>DFHAPRC</td>
<td>WAIT_OLDC</td>
<td>MISC System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>SMPRESOS</td>
<td>(none)</td>
<td>DFHSMSY</td>
<td>SUSPEND TIMER</td>
<td>MISC System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>SMSYSTEM</td>
<td>(none)</td>
<td>DFHSMSY</td>
<td>SUSPEND TIMER</td>
<td>MISC System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>SMSYRE</td>
<td>SMVA_ECB</td>
<td>DFHSMSY</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>SUCNSOLE</td>
<td>WTO</td>
<td>DFHSUWT</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>TCLASS</td>
<td>No, No</td>
<td>tclass name</td>
<td>DFHXMAT</td>
<td>User</td>
<td>Checking log stream status on page 216</td>
<td></td>
</tr>
<tr>
<td>TCLASS</td>
<td>Yes, Yes</td>
<td>tclass name</td>
<td>DFHXMCL</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TCP_NORM</td>
<td>DFHZDSP</td>
<td>DFHZDSP</td>
<td>WAIT_OLDW</td>
<td>See note 11 on page 124</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TCP_SHUT</td>
<td>DFHZDSP</td>
<td>DFHZDSP</td>
<td>WAIT_OLDW</td>
<td>MISC System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>TCTVCECB</td>
<td>ZC_ZGRP</td>
<td>DFHZGRP</td>
<td>WAIT_OLDC</td>
<td>MISC System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>TDEPLOCK</td>
<td>No, No</td>
<td>transient data queue name</td>
<td>DFHTDA</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TD_INIT</td>
<td>No, No</td>
<td>DCT</td>
<td>DFHTDA</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TD_INIT</td>
<td>No, No</td>
<td>DCT</td>
<td>DFHTDA</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TD_READ</td>
<td>No, No</td>
<td>transient data queue name</td>
<td>DFHTDB</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TIEXPIRY</td>
<td>DS_NUDGE</td>
<td>DFHTISR</td>
<td>SUSPEND TIMER</td>
<td>System only</td>
<td>CICS system task waits on page 140</td>
<td></td>
</tr>
<tr>
<td>TRANDEF</td>
<td>Yes, Yes</td>
<td>transaction id</td>
<td>DFHXMDD DFHXMQD</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TSAUX</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TSBUFFER</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TSEXEXTEND</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>TSIO</td>
<td>No, No</td>
<td>(none)</td>
<td>DFHTSAM</td>
<td>WAIT_MVS IO</td>
<td>User</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 7. How tasks are made to wait 121
<table>
<thead>
<tr>
<th>Resource type</th>
<th>Purge status</th>
<th>Resource name</th>
<th>Suspending module</th>
<th>DSSR call and WLM wait type</th>
<th>Task</th>
<th>Where to look next</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSIOWAIT</td>
<td>Yes, Yes</td>
<td>DFHTTEMP</td>
<td>DFHTSDM</td>
<td>WAIT_MVS IO</td>
<td>System only</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>TSOOL</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>TSQUEUE</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>TSSHARED</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSSH</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>TSSTRING</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>TSWBUFFR</td>
<td>Yes, Yes</td>
<td>temporary storage queue name</td>
<td>DFHTSWQ</td>
<td>SUSPEND LOCK</td>
<td>User</td>
<td>&quot;Investigating temporary storage waits on page 69&quot;</td>
</tr>
<tr>
<td>UDSA</td>
<td>Yes, Yes</td>
<td>(none)</td>
<td>DFHSMSQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;Investigating storage waits on page 67&quot;</td>
</tr>
<tr>
<td>USERWAIT</td>
<td>Yes, Yes or No, No</td>
<td>Supplied by application</td>
<td>DFHEIQSK</td>
<td>WAIT_MVS MISC WAIT_OLDW MISC</td>
<td>User</td>
<td>&quot;Task control waits on page 130&quot;</td>
</tr>
<tr>
<td>USERWAIT</td>
<td>Yes, Yes</td>
<td>CDB2TIME</td>
<td>DFHD2EX2</td>
<td>WAIT_OLDW MISC</td>
<td>System</td>
<td>CICS DB2 Guide</td>
</tr>
<tr>
<td>USERWAIT</td>
<td>Yes, Yes</td>
<td>DB2START</td>
<td>DFHD2EX2</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>CICS DB2 Guide</td>
</tr>
<tr>
<td>WBALIAS</td>
<td>No, No</td>
<td>Target Transid</td>
<td>DFHWBXN</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;CICS Web waits on page 142&quot;</td>
</tr>
<tr>
<td>WEB_ECB</td>
<td></td>
<td>DFH_STATE_TOKEN</td>
<td>DFHWBST</td>
<td>WAIT_MVS TIMER</td>
<td>System or User</td>
<td></td>
</tr>
<tr>
<td>XRGETMSG</td>
<td></td>
<td>message queue name</td>
<td>DFHWMQG</td>
<td>WAIT_MVS See note 7 on page 123</td>
<td>System only</td>
<td>&quot;CICS system task waits on page 146&quot;</td>
</tr>
<tr>
<td>XRPUTMSG</td>
<td>Yes, Yes</td>
<td>message queue name</td>
<td>DFHWMQP</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>&quot;XRF alternate system waits on page 139&quot;</td>
</tr>
<tr>
<td>ZC</td>
<td>Yes, No</td>
<td>DFHZCRQ1</td>
<td>DFHZCRQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
<tr>
<td>ZC</td>
<td>Yes, No</td>
<td>DFHZEMW1</td>
<td>DFHZEMW</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
<tr>
<td>ZC</td>
<td>Yes, No</td>
<td>DFHZIS11</td>
<td>DFHZIS1</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
<tr>
<td>ZC</td>
<td>Yes, No</td>
<td>DFHZRAQ1</td>
<td>DFHZRAQ</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
<tr>
<td>ZC</td>
<td>Yes, No</td>
<td>DFHZRAR1</td>
<td>DFHZRAR</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
<tr>
<td>ZC_ZCGRP</td>
<td></td>
<td>ZSLSECB</td>
<td>DFHZCGRP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td>&quot;VTAM terminal control waits on page 133&quot;</td>
</tr>
</tbody>
</table>
Table 9. Resources on which a suspended task might be waiting (continued)

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Purge status</th>
<th>Resource name</th>
<th>Suspending module</th>
<th>DSSR call and WLM wait type</th>
<th>Task</th>
<th>Where to look next</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZC_ZGCH</td>
<td>No, No</td>
<td>CHANGECB</td>
<td>DFHZGCH</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZC_ZGIN</td>
<td>No, No</td>
<td>INQ_ECB_</td>
<td>DFHZGIN</td>
<td>WAIT_MVS MISC</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZC_ZGRP</td>
<td>PSINQECB</td>
<td>DFHZGRP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZC_ZGRP</td>
<td>PSOP1ECB</td>
<td>DFHZGRP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZC_ZGRP</td>
<td>PSOP2ECB</td>
<td>DFHZGRP</td>
<td>WAIT_MVS MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZC_ZGUB</td>
<td>PSUNBECB</td>
<td>DFHZGUB</td>
<td>WAIT_OLDC MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZARQ1</td>
<td>DFHZARQ</td>
<td>SUSPEND See note 12</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZARL1</td>
<td>DFHZARL</td>
<td>SUSPEND See note 12</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZARL4</td>
<td>DFHZARL</td>
<td>SUSPEND See note 12</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZARR1</td>
<td>DFHZARR1</td>
<td>SUSPEND See note 12</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZARER</td>
<td>DFHZARER</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZERH1</td>
<td>DFHZERH</td>
<td>SUSPEND CONV</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZERH2</td>
<td>DFHZERH</td>
<td>SUSPEND CONV</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCIOWAIT</td>
<td>Yes, No</td>
<td>DFHZERH3</td>
<td>DFHZERH</td>
<td>SUSPEND CONV</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCZGET</td>
<td>Yes, No</td>
<td>DFHZARL2</td>
<td>DFHZARL</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCZNAC</td>
<td>Yes, No</td>
<td>DFHZARL3</td>
<td>DFHZARL</td>
<td>SUSPEND MISC</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZCZNAC</td>
<td>Yes, No</td>
<td>DFHZERH4</td>
<td>DFHZERH</td>
<td>SUSPEND CONV</td>
<td>User</td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZXQOWAIT</td>
<td>LIST</td>
<td>DFHZXQO</td>
<td>WAIT_OLDW MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
<tr>
<td>ZXQOWAIT</td>
<td>LIST</td>
<td>DFHZXST</td>
<td>WAIT_OLDW MISC</td>
<td>System only</td>
<td></td>
<td>“VTAM terminal control waits” on page 133</td>
</tr>
</tbody>
</table>

Notes:
1. The MVS workload manager monitoring environment is set to STATE=IDLE when either:
   - A conversational task is waiting for terminal input from its principal facility, or
   - A CICS system task is waiting for work.
2. These waits are used by the CICS/ESA Front End Programming Interface and are not discussed in this manual.
3. If the task is waiting on resource type ALLOCATE, the current MVS workload manager monitoring environment is set to STATE=WAITING and either:
   • RESOURCE=SESS_LOCALMVS if the session being waited on is a session with another CICS region in the same local MVS image.
   • RESOURCE=SESS_SYSPLEX if the session being waited on is a session with a CICS region in another MVS image in the same sysplex.
   • RESOURCE=SESS_NETWORK if the session being waited on is an ISC session which may, or may not, be in the same MVS image.

4. The task is waiting for a deployed jar file (DJAR) to become available for use.

5. The task is waiting for a Corbaserver and its associated DJARs to become available for use.

6. If there is a terminal associated with the task.

7. If the task is waiting on resource type IRLINK, the current MVS 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 MVS monitoring environment. The SWITCHED column in this report can contain the following values:
   • LOCALMVS - the communicating CICS region is on the same local MVS image.
   • SYSPLEX - the communicating CICS region is on another MVS image in the same sysplex.

8. “xxx” is literal.

9. The task has not yet started, because the system is at MXT.

10. The task is waiting in the pipeline for another task to complete. These tasks are connected through the Request Stream (RZ) component. The tasks could be using MRO, for example as part of a Web service atomic transaction that is registering with a coordination service and is waiting for a response. The tasks could be local to the CICS region, in which case MRO is not involved even though the tasks are still using Request Streams.

11. The task has not yet started because it is being held for transaction class purposes.

12. If the task is waiting on resource type ZCIOWAIT, the current MVS workload manager monitoring environment is set to either:
   • 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 MVS monitoring environment. The SWITCHED column in this report can contain the following values:
     • LOCALMVS - the communicating CICS region is on the same local MVS image.
     • SYSPLEX - the communicating CICS region is on another MVS image in the same sysplex.
     • NETWORK - the communicating CICS region is in the VTAM network, which may, or may not be, in the same MVS image.

13. These waits occur in the partner domain, and are currently only used by the Link3270 bridge. The Link3270 bridge consists of two tasks, either of which could be waiting.
• The linking task issues a start for the user transaction running under the bridge, and waits for a response from this user transaction. This response would either be the user transaction completing or requesting more information.

• The user task issues the wait if it is requesting further information from the caller. This would be a conversational wait.

The way to distinguish between the two tasks is that the user task has a bridge facility associated with it.

### Dispatcher waits

There are five reasons why CICS dispatcher might cause tasks to wait, and the resource names or resource type associated with these are:

• JVM_POOL
• OPENPOOL
• OPEN_DEL
• DSTSKDEF
• SOSMVS

#### Resource name JVM_POOL

When a task first needs a J8 or J9 mode open TCB, the dispatcher domain attempts to find a free TCB from the JVM pool. If there is not a free J8 or J9 mode TCB, and the number of open TCBs in the JVM pool is less than MAXJVMTCBS, CICS attaches a new TCB, and allocates this to the requesting task.

However, if the number of J8 and J9 TCBs in the pool is at the limit set by MAXJVMTCBS, 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 MAXJVMTCBS limit is raised, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

#### Resource name OPENPOOL

When a task first needs an L8 mode open TCB, the dispatcher domain attempts to find a free TCB of this mode with the correct subspace attributes. If there is not a free L8 mode TCB associated with a matching subspace, CICS:

• Attaches a new L8 mode TCB of the required subspace if the number of open TCBs in the L8 open TCB pool is less than MAXOPENTCBS, and allocates the new TCB to the requesting task.

• Detaches a free open L8 mode TCB associated with a different subspace (if there is one available and MAXOPENTCBS limit has been reached), attaches a new L8 mode TCB, and allocates this new TCB to the requesting task.¹

However, if neither of these options is available, 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 the MAXOPENTCBS limit is raised, the task at the front of the queue is resumed, and the open TCB allocation process is retried.

¹ This process is referred to as TCB stealing: deleting a free TCB of one type in order to attach one of a different type.
Resource name OPEN_DEL

If your task is waiting on a resource name of OPEN_DEL, 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 terminate so that dispatcher can attach a new one.

Your task needs an open TCB, but no suitable TCB is available, and a new TCB cannot be attached because the system is constrained by the MAXOPENTCBS limit. In this situation, CICS selects a currently idle TCB for termination, to allow the task to attach a TCB of the required type. However, the attach cannot proceed until the deleted TCB's termination is complete, otherwise the number of open TCBs in the L8 pool would temporarily exceed MAXOPENTCBS.

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 could be waiting for a dump to complete, for example.

Resource type SOSMVS

The number of open TCBs (J8 or J9 mode TCB) in the JVM pool is constrained by the MAXJVMTCBS system initialization parameter. If you set a MAXJVMTCBS limit that is too high, CICS might attempt to create too many JVMs for the available MVS storage, resulting in an MVS storage constraint.

CICS has a storage monitor for MVS storage, which notifies it when MVS storage is constrained or severely constrained, so that it can take short-term action to reduce the number of JVMs in the JVM pool. As JVMs make requests for MVS storage, the storage monitor checks whether the availability of MVS storage has dropped below a pre-set threshold of 40MB, and notifies CICS when this is the case. The storage monitor also notifies CICS if the availability of MVS storage has become so low that MVS storage requests can only be satisfied from a pre-set MVS storage cushion of 20MB.

When the storage cushion is breached and so MVS storage is severely constrained, CICS temporarily prevents the creation of new JVMs for incoming requests, and behaves as though the MAXJVMTCBS limit has been reached and the JVM pool is full. In this situation, if the storage monitor is still receiving requests from CICS to create JVMs, it queues any such requests that cannot obtain sufficient MVS storage. These requests are suspended with a resource name of SOSMVS.

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

This resource type is used instead of CDB2TCB when CICS is connected to DB2 Version 6 or later.

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 ( ) THREADLIMITgetResource number) command. You can increase the number of threads available for the pool with a SET DB2CONN THREADLIMIT(nnn) command. If you increase the THREADLIMIT value, CICS posts tasks to retry acquisition of a thread.

### Resource type CDB2TCB

This resource type is used instead of CDB2CONN when CICS is connected to DB2 Version 5 or earlier.

The task is waiting for a CICS DB2 subtask to become available. This indicates that the TCBLIMIT value has been reached and that all TCBs are in use and running 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 TCB has been acquired.

You can increase the number of TCBs permitted with a SET DB2CONN TCBLIMIT command. If you increase the TCBLIMIT value, CICS posts tasks to retry acquisition of a DB2 subtask.

### Resource type DB2

Used when CICS is connected to DB2 Version 5 or earlier. The task is waiting for the CICS DB2 request to be completed. If forcepurged in this state, the task abends and backout occurs, and the thread TCB is detached, causing DB2 also to back out.

### 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.

DBCTL waits

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

- You have attempted to connect to DBCTL using the CICS-supplied transaction CDBC, but the connection process has failed to complete.
- You have a user task in a wait state, and you have found that it is waiting on resource type DBCTL, with resource name DLSUSPND.
- You have attempted to disconnect from DBCTL using the CICS-supplied transaction CDBC, but the disconnection process has failed to complete.

Each of these types of problem is dealt with in turn.

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 simply 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 the IMS Diagnosis Guide and Reference manual for guidance about debugging the problem.

A user task is waiting on resource type DBCTL

If you have found that a user task is waiting on a resource type of DBCTL, and resource name DLSUSPND, the task has made a DL/I request. It has been suspended by CICS while the request is serviced by DBCTL. If the task has not resumed, the request has not been completed, for some reason.

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 the IMS Diagnosis Guide and Reference manual 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 may provide further information about the cause of the wait. The MVS system console may 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.

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 may be purged if the log stream is not DFHLOG, the primary system log.

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 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.

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 the CICS Application Programming Reference.

### 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 10.

<table>
<thead>
<tr>
<th>Resource name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS</td>
<td>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.</td>
</tr>
<tr>
<td>LIST</td>
<td>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.</td>
</tr>
<tr>
<td>SINGLE</td>
<td>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.</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 57.)

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

If you find an invalid address: It is possible that a storage overlay is the cause of the wait problem. If you suspect this to be the case, turn to Chapter 11, “Dealing with storage violations,” on page 193 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 84.
- 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 91.

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.
VTAM terminal control waits

VTAM terminal control waits are associated with the following resource types:

- ZC
- ZC_ZCGRP
- ZC_ZGCH
- ZC_ZGIN
- ZC_ZGRP
- ZC_ZGUB
- ZCIOWAIT
- ZCZGET
- ZCZNAC
- ZXQOWAIT
- ZXSTWAIT

The implication of waits on any of these VTAM terminal resource types are as follows:

Resource type ZC

If your task is waiting on a resource name of DFHZCRQ1, it is waiting for I/O to complete. The task is attempting to complete one of the following:

- RESETSR
- A send synchronous data flow
- A send asynchronous command
- SESSIONC.

The task waits for the time specified in the 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 VTAM 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 VTAM CHANGE ENDAFFIN macro to complete.

Resource type ZC_ZGIN

DFHZGIN issues the VTAM INQUIRE macro and waits until VTAM completes execution of this request.
Resource type ZC_ZGRP, resource name PSINQECB
If the task is waiting on resource name PSINQECB, this means that DFHZGRP has issued the VTAM macro INQUIRE PERSESS during VTAM persistent session restart, or during the reopening of the VTAM ACB, and is waiting for a response from VTAM. The wait expires after 5 minutes if VTAM does not respond.

Resource type ZC_ZGRP, resource name PSOP2ECB
If the task is waiting for resource name PSOP2ECB, this means that DFHZGRP has issued the VTAM macro OPNDST RESTORE during emergency restart, and is waiting for a response from VTAM. The wait expires after 5 minutes if VTAM does not respond.

Resource type ZC_ZGUB
DFHZGUB issues ten VTAM CLSDST or TERMSESS macros during persistent sessions restart. It waits for an RPL to become free for VTAM to post the VTAM exit. The wait expires after 5 minutes if VTAM does not respond.

Resource type ZCIOWAIT
Suspends on resource type ZCIOWAIT occur when the task is waiting for some terminal I/O. Once the expected I/O event occurs, the task is resumed.

Resource type ZCZGET
If your task is waiting on a resource name of DFHZARL2, it is suspended by module DFHZARL which deals with application request logic for 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 the CICS Resource Definition Guide and the CICS Intercommunication Guide.

If you otherwise have a problem that you have identified as an interregion or an intersystem communication wait, investigate it as for terminal waits. This is dealt with in "Investigating terminal waits" on page 57.
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.

IIOP waits

A request receiver DFHIIRR task suspends with resource type IIRR and resource name SOCBNOTI when it is waiting for input from the client or a reply from the request processor and the TCP/IP connection is still open. It is resumed by a NOTIFY gate when IIRR is told there is another request from the sockets domain or a reply has come in from the request streams domain.

SOCFNOTI is similar to SOCBNOTI, except that it also indicates that a client has sent parts of a GIOP fragment but has not yet sent the final fragment.

A request processor DFHIIRP task suspends with resource type IIRP and resource name NOTI when it is waiting for requests or replies. It is resumed by a NOTIFY gate when IIRP is told there is another request or reply from the request streams domain.

Transient data waits

Tasks issuing requests to read and write to transient data destinations can be made to wait for several different reasons. The reasons depend on the type of request being made, and whether the task is attempting to access an extrapartition or an intrapartition queue.

The resource types that might be associated with the wait are:
- TD_INIT
- TDEPLOCK
- TDIPLOCK
- ENQUEUE
- TD_READ
- Any_MBCB
- Any_MRCB
- MBCB_xxx
- MRCB_xxx

The resource name is the transient data queue name, except in the case of TD_INIT, whose resource name is DCT.

Resource type TD_INIT—waits during initialization processing

A second stage PLT program being executed during system initialization can issue a request for a resource that is not yet available, because the component that services the request has not yet been initialized. If the program issues a transient data request that cannot yet be serviced, it is suspended on a resource type of TD_INIT with a resource name of DCT.
You are unlikely to see any evidence for this type of wait, unless you have trace running during initialization with DS level-1 tracing selected. An error at this stage would be likely to cause CICS to stall (see “CICS has stalled during initialization” on page 106), or to terminate abnormally.

**Resource type TDEPLOCK—waits for transient data extrapartition requests**

If you have a task suspended on resource type TDEPLOCK, with a resource name corresponding to a transient data queue name, the task has issued a request against an extrapartition transient data queue. Another task is already accessing the same queue, and the waiting task cannot resume until that activity is complete.

If the wait is prolonged, it could be for either of these reasons:

- It is necessary for a task to change 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 executes synchronously with tasks requesting its services. This means that any task invoking a QSAM service must wait until the QSAM processing is complete. If, for any reason, QSAM enters an extended wait, the requesting task also experiences an extended wait.

  The possibility of an extended wait arises whenever 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 attempting 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, TD_READ, Any_MBCB, Any_MRCB, MBCB_xxx, and MRCB_xxx**

If your task is waiting on any of the resource types TDIPLOCK, ENQUEUE, 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**

If you have a task suspended on resource type TDIPLOCK, with a resource name corresponding to a transient data queue name, the task has issued a request against an intrapartition transient data queue. Another task is already accessing the same queue, and the waiting task cannot resume until that activity is complete.

If tasks frequently get suspended on resource type TDIPLOCK, you need to determine which other transactions use the same intrapartition destination. You might then consider redefining the intrapartition destinations.

You can find further guidance information about the constraints that apply to tasks writing to intrapartition destinations in the [CICS Application Programming Guide](#). For more details of the properties of recoverable transient data queues, see the [CICS Resource Definition Guide](#).
Resource type 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 72. 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 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” and “Resource type MRCB_xxx”).

The reason for this type of wait is best illustrated by example, as follows:

1. Task #1 issues a transient data request that requires access to an intrapartition queue. Before the request can be serviced, task #1 must be assigned a transient data I/O buffer that is not currently being used by any other task. I/O buffers each contain a copy of a control interval (CI) from a data set. Each CI contains records that correspond to elements in an intrapartition queue. A search is made to see if the CI required for task #1 is already in one of the I/O buffers. If it is, that I/O buffer can be used to service the request made by task #1, and no VSAM I/O is involved. If it is not, task #1 is allocated any buffer, so the required CI can be read in. The current contents of the buffer is overwritten. An I/O buffer can have a R/O (read only) status or a R/W (read/write) status. If the buffer that is allocated to task #1 has R/W status, it contains a copy of a CI that has been updated by some other task, but not yet written back to the data set. Before the buffer can be used by task #1, the CI it contains must be preserved by writing it back to the data set.

2. A request now arrives from task #2, and the request requires the CI that is currently being written to the data set. No two buffers can contain the same CI, so task #2 is made to wait on resource type MRCB_xxx until the outcome of the VSAM I/O is known.
   If VSAM I/O was successful, task #2 is resumed and assigned some other I/O buffer.
   If VSAM I/O was unsuccessful, task #2 can use the I/O buffer that already contains the CI it needs.
XRF alternate system waits

The XRF takeover process involves several operations. Before each operation can be started, one or more events must have completed. For example:

- Terminals with backup sessions can be switched while the active system is running, provided the CICS availability manager (CAVM) has initiated a takeover.
- Passively-shared data sets must not be opened until it is known that the active system has terminated.

Note: This is the only way an alternate system can be sure that no more data will be written by an active system.

- Resource managers, such as transient data, temporary storage, and database recovery control (DBRC), rely on the time-of-day clock providing them with a nondecreasing value to ensure the proper management of their resources. The alternate system must not restart a resource manager until the alternate time-of-day clock has been synchronized with the active time-of-day clock.

A system task that issued a takeover request to CAVM waits on the ECB WCSTCECB in the CAVM static control block (DFHWCGPS) until CAVM has decided to accept or reject the request. The DFHKC TYPE=WAIT and DCI=SINGLE requests are issued in DFHWSRTR. The CAVM TCB posts WCSTCECB in either DFHWSTKV (the normal case) or DFHWSSOF (CAVM failure).

The following ECBs each represent an event. The ECBs are located in the static storage for DFHXRP. The ECBs and the events are:

- XRSTIECB—the CAVM has initiated a takeover.
- XRSIAECB—the alternate system is now the incipient active system.
- XRSTCECB—the active system is known to have terminated.
- XRSSSECB—the time-of-day clock is synchronized with active system sign off.
- XRSSTECB—the time-of-day clock is synchronized with active system termination.

XRSTIECB

This ECB is posted by DFHXRA, following a successful call to the CAVM to initiate takeover. Once the ECB has been posted, DFHXRA attaches a system transaction to initiate the switch of terminals with backup sessions. DFHXRA is called from either the surveillance task (DFHXRSP), or the console communication task (DFHXRCP). No tasks wait for XRSTIECB to be posted.

XRSIAECB

The XRSIAEB ECB is posted by DFHXRA, following notification by the CAVM that an alternate system is now the incipient active system. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSIAECB to be posted.

XRSTCECB

The XRSTCECB ECB is posted by DFHXRA, following notification by the CAVM that an active system has terminated. There can be a delay in posting the ECB if:

- An SDUMP is being taken as part of the active system termination process.
- In a 2-CPC environment, the active CPC has failed, and the operator failed to reply to the messages sent to the console.
DFHXRA is called from the surveillance task (DFHXRSP). Only one task, the system initialization task (DFHSII1), waits for XRSTCECB to be posted. When the ECB is posted, DFHSII1 opens the restart data set, for DFHRC use as well as for DFHCC use, and then calls DFHXRA to post the XRSRAECB.

XRSRAECB

The XRSRAECB ECB is posted by DFHXRA once the restart data set has been opened, for DFHRC use as well as for DFHCC use. DFHXRA is called from the system initialization task (DFHSII1). Two tasks wait for XRSRAECB to be posted:

- The transient data recovery task (DFHTDRP) initializes the entry for the CXRF queue before waiting for XRSRAECB to be posted. When the ECB is posted, DFHTDRP resumes emergency restart processing.
- The terminal control recovery task (DFHTCRP) drains its tracking queue before waiting for XRSRAECB to be posted. When the ECB is posted, DFHTCRP resumes emergency restart processing.

XRSSSECB

The XRSSSECB ECB is posted by DFHXRA following notification by the CAVM that the time-of-day clock is synchronized with active sign off. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSSSECB to be posted.

XRSSTECB

The XRSSTECB ECB is posted by DFHXRA, following notification by the CAVM that the time-of-day clock is synchronized with respect to active system termination. There may be a delay in posting the ECB if the time indicated by the active system time-of-day clock is significantly ahead of that indicated by the alternate system time-of-day clock. DFHXRA is called from the surveillance task (DFHXRSP).

Only the system initialization task, DFHSII1, waits for XRSSTECB to be posted.

You are only likely to find either of the CICS-supplied transactions CEDA or CESN waiting on a resource type of XRPUTMSG, and only during XRF takeover by the alternate CICS system. It can indicate either of these conditions:

- Data that is required by the transactions is held on a volume subject to MVS RESERVE locking, and another job currently has the lock.
- There is an error in the CICS availability manager.

If it seems clear that MVS RESERVE locking is not implicated, refer the problem to the IBM Support Center.

CICS system task waits

From an analysis of trace, you could have evidence that a CICS system task is in a wait state. You might have seen the task suspended on a SUSPEND call to the dispatcher, but with no corresponding RESUME call. Alternatively, by looking at the dispatcher task summary in a formatted CICS system dump, you might see that a CICS system task is waiting.

Note: You cannot get online information about waiting system tasks from CEMT 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 11 shows the points at which FEPI issues CICS waits:

**Table 11. FEPI waits**

<table>
<thead>
<tr>
<th>Resource name</th>
<th>Resource type</th>
<th>Wait type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEPI_RQE</td>
<td>ADAPTER</td>
<td>WAIT_MVS</td>
<td>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.</td>
</tr>
<tr>
<td>SZRDP</td>
<td>FEPRM</td>
<td>WAIT_MVS</td>
<td>Issued in the FEPI Resource Manager when it has no work to do. Ends when work arrives (from either the FEPI adapter or a VTAM exit).</td>
</tr>
</tbody>
</table>

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, resource name LOGMOVE
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.

Resource type RMUOWOBJ, resource name EXISTENC
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.
Chapter 8. Dealing with loops

A loop is a sequence of instructions that is executed repetitively. Loops that are coded into applications must always be guaranteed to terminate, because otherwise you could get any of the symptoms of loops described in Chapter 2, "Classifying the problem," on page 7.

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:

- "What sort of loop is indicated by the symptoms?"
- "Investigating lock manager waits" on page 93
- "Investigating loops that are not detected by CICS" on page 157
- "What to do if you cannot find the reason for a loop" on page 158

If you find that the looping code is in one of your applications, you need to 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 DFHMMN0004.

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 15** gives an example of code containing a simple tight loop.
CICS can detect some looping tasks by comparing the length of time the tasks have been running with the runaway time interval, ICVR, that you code in the system initialization table. If a task runs for longer than the interval you specify, CICS regards it as “runaway” and causes it to abend with an abend code of AICA. However, in some cases, CICS requests that are contained in the looping code can cause the timer to be reset. Not every CICS request can do this; it can only happen if the request can cause the task to be suspended. Thus, if the looping code contains such a request, CICS cannot detect that it is looping.

The properties of the different types of loop, and the ways you can investigate them, are described in the sections that follow.

**Tight loops and non-yielding loops**

Tight loops and non-yielding loops are both characterized by the fact that the looping task can never be suspended within the limits of the loop. This makes them detectable by CICS, which compares the time they have been running continually with the runaway time interval, ICVR, that you code in the system initialization table. If the tasks run for longer than the interval you specify, CICS regards them as “runaway” and causes them to abend with an abend code of AICA.

**Note:** If you make the ICVR value equal to 0, runaway task detection is disabled. Runaway tasks can then cause the CICS 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 desired 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 16 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.

```plaintext
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 16. 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 146 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 106. 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 17 on page 146 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.
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 13.

If you suspect that you have a yielding loop, turn to "Investigating loops that are not detected by CICS" on page 157 for further guidance.

Investigating loops that cause transactions to abend with abend code AICA

If the loop causes a transaction to abend with abend code AICA, it must either be a tight loop or a non-yielding loop. You do not need to find which type you have, although this is likely to be revealed to you when you do your investigation.

Both a tight loop and a non-yielding loop are characterized by being confined to a single user program. You should know the identity of the transaction to which the program belongs, because it is the transaction that abended with code AICA when the runaway task was detected.

The documentation you need

When investigating loops that cause transactions to abend AICA, you need the CICS system dump accompanying the abend. System dumping must be enabled for dump code AICA.

You can use the system dump to find out:
• Whether the loop is in your user code or in CICS code
• If the loop is in your user code, the point at which the loop was entered.

It is also useful to have trace running, as trace can help you to identify the point in your program where looping started. If you have a non-yielding loop, it can probably also show you some instructions in the loop.

A tight loop is unlikely to contain many instructions, and you might be able to capture all the evidence you need from the record of events in the internal trace table. A non-yielding loop may contain more instructions, depending on the EXEC CICS commands it contains, but you might still be able to capture the evidence you
need from the record of events in the internal trace table. If you find that it is not big enough, direct tracing to the auxiliary trace destination instead.

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:
1. Select level-1 special tracing for AP domain, and for the EXEC interface program (EI).
2. Select special tracing for just the task that has the loop, and disable tracing for all other tasks by turning the master system trace flag off.

You can find guidance about setting up these tracing options in Chapter 15, “Using traces in problem determination,” on page 225.

Now start the task, and wait until it abends AICA. Format the CICS system dump with formatting keywords KE and TR, to get the kernel storage areas and the internal trace table. (See “Formatting system dumps” on page 285.) You now have the documentation you need to find the loop.

Looking at the evidence

Look first at the kernel task summary. The runaway task is flagged “**YES**” in the ERROR column. The status of the task is shown as ****Running****.

Now use the kernel task number for the looping task to find its linkage stack. If a user task is looping, DFHAPLI, a transaction manager program, should be near the top of the stack. You are likely to find other CICS modules at the top of the stack that have been invoked in response to the abend. For example, those associated with taking the dump. If you find any program or subroutine above DFHAPLI that has not been invoked in response to the error, it is possible that CICS code, or the code of another program, has been looping. If you find that the loop is within CICS code, 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

There are two approaches to identifying loops in user programs. You can use the trace table, or you can look in the transaction dump.

Using the trace table

Go to the last entry in the internal trace table, and work backward until you get to an entry for point ID AP 1942. The trace entry should have been made when recovery was entered after the transaction abended AICA. Make a note of the task number, so you can check that any other trace entries you read relate to the same abended task.

The entries preceding AP 1942 should have been made either just before the loop was entered (for a tight loop), or within the loop itself (for a non-yielding loop). Watch in particular for trace entries with the point ID AP 00E1. These are made on entry to the EXEC interface program (DFHEIP) whenever your program issues an EXEC CICS command, and again on exit from the EXEC interface program. Field B gives you the value of EIBFN, which identifies the specific command that was issued. The table below contains a list of EIBFN values and their meanings.
In the column of the EIBFN codes table headed **Type**, API indicates that the command is described in the [CICS Application Programming Reference](#) and SPI indicates that it is described in the [CICS System Programming Reference](#). SPI does **not** indicate that the special (SP) translator option is required for this command. FEPI indicates commands used by the CICS/ESA Front End Programming Interface. They are described in the CICS Front End Programming Interface User’s Guide.

**Table 12. EIB Field Name Values & Types**

<table>
<thead>
<tr>
<th>EIBFN value</th>
<th>Command</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'0202'</td>
<td>ADDRESS</td>
<td>API</td>
</tr>
<tr>
<td>X'0204'</td>
<td>HANDLE CONDITION</td>
<td>API</td>
</tr>
<tr>
<td>X'0206'</td>
<td>HANDLE AID</td>
<td>API</td>
</tr>
<tr>
<td>X'0208'</td>
<td>ASSIGN</td>
<td>API</td>
</tr>
<tr>
<td>X'020A'</td>
<td>IGNORE CONDITION</td>
<td>API</td>
</tr>
<tr>
<td>X'020C'</td>
<td>PUSH</td>
<td>API</td>
</tr>
<tr>
<td>X'020E'</td>
<td>POP</td>
<td>API</td>
</tr>
<tr>
<td>X'0210'</td>
<td>ADDRESS SET</td>
<td>API</td>
</tr>
<tr>
<td>X'0402'</td>
<td>RECEIVE</td>
<td>API</td>
</tr>
<tr>
<td>X'0404'</td>
<td>SEND</td>
<td>API</td>
</tr>
<tr>
<td>X'0406'</td>
<td>CONVERSE</td>
<td>API</td>
</tr>
<tr>
<td>X'0408'</td>
<td>ISSUE EOS</td>
<td>API</td>
</tr>
<tr>
<td>X'040A'</td>
<td>ISSUE COPY</td>
<td>API</td>
</tr>
<tr>
<td>X'040C'</td>
<td>WAIT TERMINAL</td>
<td>API</td>
</tr>
<tr>
<td>X'040E'</td>
<td>ISSUE LOAD</td>
<td>API</td>
</tr>
<tr>
<td>X'0410'</td>
<td>WAIT SIGNAL</td>
<td>API</td>
</tr>
<tr>
<td>X'0412'</td>
<td>ISSUE RESET</td>
<td>API</td>
</tr>
<tr>
<td>X'0414'</td>
<td>ISSUE DISCONNECT</td>
<td>API</td>
</tr>
<tr>
<td>X'0416'</td>
<td>ISSUE ENDOOUTPUT</td>
<td>API</td>
</tr>
<tr>
<td>X'0418'</td>
<td>ISSUE ERASEAUP</td>
<td>API</td>
</tr>
<tr>
<td>X'041A'</td>
<td>ISSUE ENDFILE</td>
<td>API</td>
</tr>
<tr>
<td>X'041C'</td>
<td>ISSUE PRINT</td>
<td>API</td>
</tr>
<tr>
<td>X'041E'</td>
<td>ISSUE SIGNAL</td>
<td>API</td>
</tr>
<tr>
<td>X'0420'</td>
<td>ALLOCATE</td>
<td>API</td>
</tr>
<tr>
<td>X'0422'</td>
<td>FREE</td>
<td>API</td>
</tr>
<tr>
<td>X'0424'</td>
<td>POINT</td>
<td>API</td>
</tr>
<tr>
<td>X'0426'</td>
<td>BUILD ATTACH</td>
<td>API</td>
</tr>
<tr>
<td>X'0428'</td>
<td>EXTRACT ATTACH</td>
<td>API</td>
</tr>
<tr>
<td>X'042A'</td>
<td>EXTRACT TCT</td>
<td>API</td>
</tr>
<tr>
<td>X'042C'</td>
<td>WAIT CONVID</td>
<td>API</td>
</tr>
<tr>
<td>X'042E'</td>
<td>EXTRACT PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'0430'</td>
<td>ISSUE ABEND</td>
<td>API</td>
</tr>
<tr>
<td>X'0432'</td>
<td>CONNECT PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'0434'</td>
<td>ISSUE CONFIRMATION</td>
<td>API</td>
</tr>
<tr>
<td>X'0436'</td>
<td>ISSUE ERROR</td>
<td>API</td>
</tr>
<tr>
<td>X'0438'</td>
<td>ISSUE PREPARE</td>
<td>API</td>
</tr>
<tr>
<td>X'043A'</td>
<td>ISSUE PASS</td>
<td>API</td>
</tr>
<tr>
<td>X'043C'</td>
<td>EXTRACT LOGONMSG</td>
<td>API</td>
</tr>
<tr>
<td>X'043E'</td>
<td>EXTRACT ATTRIBUTES</td>
<td>API</td>
</tr>
<tr>
<td>X'0602'</td>
<td>READ</td>
<td>API</td>
</tr>
<tr>
<td>X'0604'</td>
<td>WRITE FILE</td>
<td>API</td>
</tr>
<tr>
<td>X'0606'</td>
<td>REWRITE</td>
<td>API</td>
</tr>
<tr>
<td>X'0608'</td>
<td>DELETE</td>
<td>API</td>
</tr>
<tr>
<td>X'060A'</td>
<td>UNLOCK</td>
<td>API</td>
</tr>
<tr>
<td>X'060C'</td>
<td>STARTBR</td>
<td>API</td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>-------</td>
</tr>
<tr>
<td>X'060E'</td>
<td>READNEXT</td>
<td>API</td>
</tr>
<tr>
<td>X'0610'</td>
<td>READPREV</td>
<td>API</td>
</tr>
<tr>
<td>X'0612'</td>
<td>ENDBR</td>
<td>API</td>
</tr>
<tr>
<td>X'0614'</td>
<td>RESETBR</td>
<td>API</td>
</tr>
<tr>
<td>X'0802'</td>
<td>WRITEQ TD</td>
<td>API</td>
</tr>
<tr>
<td>X'0804'</td>
<td>READQ TD</td>
<td>API</td>
</tr>
<tr>
<td>X'0806'</td>
<td>DELETEQ TD</td>
<td>API</td>
</tr>
<tr>
<td>X'0A02'</td>
<td>WRITEQ TS</td>
<td>API</td>
</tr>
<tr>
<td>X'0A04'</td>
<td>READQ TS</td>
<td>API</td>
</tr>
<tr>
<td>X'0A06'</td>
<td>DELETEQ TS</td>
<td>API</td>
</tr>
<tr>
<td>X'0C02'</td>
<td>GETMAIN</td>
<td>API</td>
</tr>
<tr>
<td>X'0C04'</td>
<td>FREEMAIN</td>
<td>API</td>
</tr>
<tr>
<td>X'0E02'</td>
<td>LINK</td>
<td>API</td>
</tr>
<tr>
<td>X'0E04'</td>
<td>XCTL</td>
<td>API</td>
</tr>
<tr>
<td>X'0E06'</td>
<td>LOAD</td>
<td>API</td>
</tr>
<tr>
<td>X'0E08'</td>
<td>RETURN</td>
<td>API</td>
</tr>
<tr>
<td>X'0E0A'</td>
<td>RELEASE</td>
<td>API</td>
</tr>
<tr>
<td>X'0E0C'</td>
<td>ABEND</td>
<td>API</td>
</tr>
<tr>
<td>X'0E0E'</td>
<td>HANDLE ABEND</td>
<td>API</td>
</tr>
<tr>
<td>X'1002'</td>
<td>ASKTIME</td>
<td>API</td>
</tr>
<tr>
<td>X'1004'</td>
<td>DELAY</td>
<td>API</td>
</tr>
<tr>
<td>X'1006'</td>
<td>POST</td>
<td>API</td>
</tr>
<tr>
<td>X'1008'</td>
<td>START</td>
<td>API</td>
</tr>
<tr>
<td>X'1008'</td>
<td>START BREXIT</td>
<td>API</td>
</tr>
<tr>
<td>X'100A'</td>
<td>RETRIEVE</td>
<td>API</td>
</tr>
<tr>
<td>X'100C'</td>
<td>CANCEL</td>
<td>API</td>
</tr>
<tr>
<td>X'1202'</td>
<td>WAIT EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'1204'</td>
<td>ENQ</td>
<td>API</td>
</tr>
<tr>
<td>X'1206'</td>
<td>DEQ</td>
<td>API</td>
</tr>
<tr>
<td>X'1208'</td>
<td>SUSPEND</td>
<td>API</td>
</tr>
<tr>
<td>X'1402'</td>
<td>WRITE JOURNALNUM</td>
<td>API</td>
</tr>
<tr>
<td>X'1404'</td>
<td>WAIT JOURNALNUM</td>
<td>API</td>
</tr>
<tr>
<td>X'1406'</td>
<td>WRITE JOURNALNAME</td>
<td>API</td>
</tr>
<tr>
<td>X'1408'</td>
<td>WAIT JOURNALNAME</td>
<td>API</td>
</tr>
<tr>
<td>X'1602'</td>
<td>SYNCPOINT</td>
<td>API</td>
</tr>
<tr>
<td>X'1604'</td>
<td>RESYNC ENTRYNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'1802'</td>
<td>RECEIVE MAP</td>
<td>API</td>
</tr>
<tr>
<td>X'1804'</td>
<td>SEND MAP</td>
<td>API</td>
</tr>
<tr>
<td>X'1806'</td>
<td>SEND TEXT</td>
<td>API</td>
</tr>
<tr>
<td>X'1808'</td>
<td>SEND PAGE</td>
<td>API</td>
</tr>
<tr>
<td>X'180A'</td>
<td>PURGE MESSAGE</td>
<td>API</td>
</tr>
<tr>
<td>X'180C'</td>
<td>ROUTE</td>
<td>API</td>
</tr>
<tr>
<td>X'180E'</td>
<td>RECEIVE PARTN</td>
<td>API</td>
</tr>
<tr>
<td>X'1810'</td>
<td>SEND PARTNSET</td>
<td>API</td>
</tr>
<tr>
<td>X'1812'</td>
<td>SEND CONTROL</td>
<td>API</td>
</tr>
<tr>
<td>X'1C02'</td>
<td>DUMP</td>
<td>API</td>
</tr>
<tr>
<td>X'1E02'</td>
<td>ISSUE ADD</td>
<td>API</td>
</tr>
<tr>
<td>X'1E04'</td>
<td>ISSUE ERASE</td>
<td>API</td>
</tr>
<tr>
<td>X'1E06'</td>
<td>ISSUE REPLACE</td>
<td>API</td>
</tr>
<tr>
<td>X'1E08'</td>
<td>ISSUE ABORT</td>
<td>API</td>
</tr>
<tr>
<td>X'1E0A'</td>
<td>ISSUE QUERY</td>
<td>API</td>
</tr>
<tr>
<td>X'1E0C'</td>
<td>ISSUE END</td>
<td>API</td>
</tr>
<tr>
<td>X'1E0E'</td>
<td>ISSUE RECEIVE</td>
<td>API</td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>X'1E10'</td>
<td>ISSUE NOTE</td>
<td>API</td>
</tr>
<tr>
<td>X'1E12'</td>
<td>ISSUE WAIT</td>
<td>API</td>
</tr>
<tr>
<td>X'1E14'</td>
<td>ISSUE SEND</td>
<td>API</td>
</tr>
<tr>
<td>X'2002'</td>
<td>BIF DEEDIT</td>
<td>API</td>
</tr>
<tr>
<td>X'2004'</td>
<td>DEFINE COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2006'</td>
<td>GET COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2008'</td>
<td>UPDATE COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'200A'</td>
<td>DELETE COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'200C'</td>
<td>REWIND COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'200E'</td>
<td>QUERY COUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2014'</td>
<td>DEFINE DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2016'</td>
<td>GET DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2018'</td>
<td>UPDATE DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'201A'</td>
<td>DELETE DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'201C'</td>
<td>REWIND DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'201E'</td>
<td>QUERY DCOUNTER</td>
<td>API</td>
</tr>
<tr>
<td>X'2202'</td>
<td>ENABLE PROGRAM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'2204'</td>
<td>DISABLE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'2206'</td>
<td>EXTRACT EXIT</td>
<td>SPI</td>
</tr>
<tr>
<td>X'2402'</td>
<td>ALLOCATE</td>
<td>API</td>
</tr>
<tr>
<td>X'2404'</td>
<td>ASSIGN</td>
<td>API</td>
</tr>
<tr>
<td>X'2406'</td>
<td>EXTRACT PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'2408'</td>
<td>FREE</td>
<td>API</td>
</tr>
<tr>
<td>X'240A'</td>
<td>ISSUE ABEND</td>
<td>API</td>
</tr>
<tr>
<td>X'240C'</td>
<td>CONNECT PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'240E'</td>
<td>ISSUE CONFIRMATION</td>
<td>API</td>
</tr>
<tr>
<td>X'2410'</td>
<td>ISSUE ERROR</td>
<td>API</td>
</tr>
<tr>
<td>X'2412'</td>
<td>ISSUE SIGNAL</td>
<td>API</td>
</tr>
<tr>
<td>X'2414'</td>
<td>RECEIVE</td>
<td>API</td>
</tr>
<tr>
<td>X'2416'</td>
<td>SEND</td>
<td>API</td>
</tr>
<tr>
<td>X'2418'</td>
<td>WAIT</td>
<td>API</td>
</tr>
<tr>
<td>X'241A'</td>
<td>ISSUE PREPARE</td>
<td>API</td>
</tr>
<tr>
<td>X'241C'</td>
<td>EXTRACT ATTRIBUTES</td>
<td>API</td>
</tr>
<tr>
<td>X'3002'</td>
<td>CREATE PROGRAM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3004'</td>
<td>CREATE MAPSET</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3006'</td>
<td>CREATE PARTITIONSET</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3008'</td>
<td>CREATE TRANSACTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'300A'</td>
<td>CREATE PROFILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'300C'</td>
<td>CREATE TYPETERM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'300E'</td>
<td>CREATE CONNECTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3010'</td>
<td>CREATE TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3012'</td>
<td>CREATE SESSIONS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3014'</td>
<td>CREATE FILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3016'</td>
<td>CREATE LSRPOOL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3018'</td>
<td>CREATE PARTNER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'301A'</td>
<td>CREATE TRANCLASS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'301C'</td>
<td>CREATE TDQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'301E'</td>
<td>CREATE JOURNALMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3020'</td>
<td>CREATE DB2CONN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3022'</td>
<td>CREATE DB2ENTRY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3024'</td>
<td>CREATE DB2TRAN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3026'</td>
<td>CREATE PROCSSTYPE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3028'</td>
<td>CREATE TSMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
<td>--------</td>
</tr>
<tr>
<td>X'302A'</td>
<td>CREATE ENQMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'302C'</td>
<td>CREATE REQUESTMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'302E'</td>
<td>CREATE DOCTEMPLATE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3030'</td>
<td>CREATE TCPIPSERVICE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3402'</td>
<td>DEFINE ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3404'</td>
<td>DEFINE PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3406'</td>
<td>RUN ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3408'</td>
<td>RUN ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'340E'</td>
<td>ACQUIRE PROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3410'</td>
<td>ACQUIRE ACTIVITYID</td>
<td>API</td>
</tr>
<tr>
<td>X'3412'</td>
<td>DELETE CONTAINER</td>
<td>API</td>
</tr>
<tr>
<td>X'3414'</td>
<td>GET CONTAINER</td>
<td>API</td>
</tr>
<tr>
<td>X'3416'</td>
<td>PUT CONTAINER</td>
<td>API</td>
</tr>
<tr>
<td>X'3418'</td>
<td>RESET ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'341A'</td>
<td>CHECK ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'341C'</td>
<td>CANCEL ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'341E'</td>
<td>CANCEL ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3420'</td>
<td>SUSPEND ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3422'</td>
<td>SUSPEND ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3424'</td>
<td>RESUME ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3426'</td>
<td>RESUME ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3428'</td>
<td>DELETE ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'342A'</td>
<td>LINK ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'342C'</td>
<td>LINK ACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3430'</td>
<td>RUN ACQACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3432'</td>
<td>LINK ACQACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3434'</td>
<td>SUSPEND ACQACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3436'</td>
<td>RESUME ACQACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'3438'</td>
<td>CHECK ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'343A'</td>
<td>CHECK ACQACTIVITY</td>
<td>API</td>
</tr>
<tr>
<td>X'343C'</td>
<td>RESET ACQPROCESS</td>
<td>API</td>
</tr>
<tr>
<td>X'3602'</td>
<td>DEFINE INPUT EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3602'</td>
<td>DEFINE COMPOSITE EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3604'</td>
<td>DELETE EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3608'</td>
<td>ADD SUBEVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'360A'</td>
<td>REMOVE SUBEVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'360E'</td>
<td>TEST EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3610'</td>
<td>RETRIEVE REATTACH EVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3612'</td>
<td>RETRIEVE SUBEVENT</td>
<td>API</td>
</tr>
<tr>
<td>X'3614'</td>
<td>DEFINE TIMER</td>
<td>API</td>
</tr>
<tr>
<td>X'3616'</td>
<td>DELETE TIMER</td>
<td>API</td>
</tr>
<tr>
<td>X'3618'</td>
<td>CHECK TIMER</td>
<td>API</td>
</tr>
<tr>
<td>X'361A'</td>
<td>FORCE TIMER</td>
<td>API</td>
</tr>
<tr>
<td>X'3802'</td>
<td>WEB RECEIVE</td>
<td>API</td>
</tr>
<tr>
<td>X'3804'</td>
<td>WEB SEND</td>
<td>API</td>
</tr>
<tr>
<td>X'3806'</td>
<td>WEB READ</td>
<td>API</td>
</tr>
<tr>
<td>X'3808'</td>
<td>WEB STARTBROWSE</td>
<td>API</td>
</tr>
<tr>
<td>X'380A'</td>
<td>WEB READNEXT</td>
<td>API</td>
</tr>
<tr>
<td>X'380C'</td>
<td>WEB ENDBROWSE HTTPHEADER</td>
<td>API</td>
</tr>
<tr>
<td>X'380E'</td>
<td>WEB WRITE</td>
<td>API</td>
</tr>
<tr>
<td>X'3810'</td>
<td>WEB EXTRACT</td>
<td>API</td>
</tr>
<tr>
<td>X'3814'</td>
<td>WEB RETRIEVE</td>
<td>API</td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>X'3A02'</td>
<td>INQ RRMS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'3C02'</td>
<td>DOCUMENT CREATE</td>
<td>API</td>
</tr>
<tr>
<td>X'3C04'</td>
<td>DOCUMENT INSERT</td>
<td>API</td>
</tr>
<tr>
<td>X'3C06'</td>
<td>DOCUMENT RETRIEVE</td>
<td>API</td>
</tr>
<tr>
<td>X'3C08'</td>
<td>DOCUMENT SET</td>
<td>API</td>
</tr>
<tr>
<td>X'3E0E'</td>
<td>EXTRACT TCPIP</td>
<td>API</td>
</tr>
<tr>
<td>X'3E10'</td>
<td>EXTRACT CERTIFICATE</td>
<td>API</td>
</tr>
<tr>
<td>X'4202'</td>
<td>INQUIRE AUTINSTMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4210'</td>
<td>DISCARD AUTINSTMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4402'</td>
<td>INQUIRE PARTNER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4410'</td>
<td>DISCARD PARTNER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4602'</td>
<td>INQUIRE PROFILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4610'</td>
<td>DISCARD PROFILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4802'</td>
<td>ENTER TRACENUM</td>
<td>API</td>
</tr>
<tr>
<td>X'4804'</td>
<td>MONITOR</td>
<td>API</td>
</tr>
<tr>
<td>X'4A02'</td>
<td>ASKTIME ABSTIME</td>
<td>API</td>
</tr>
<tr>
<td>X'4A04'</td>
<td>FORMATTIME</td>
<td>API</td>
</tr>
<tr>
<td>X'4C02'</td>
<td>INQUIRE FILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4C04'</td>
<td>SET FILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4C10'</td>
<td>DISCARD FILE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4E02'</td>
<td>INQUIRE PROGRAM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4E04'</td>
<td>SET PROGRAM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'4E10'</td>
<td>DISCARD PROGRAM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5002'</td>
<td>INQUIRE TRANSACTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5004'</td>
<td>SET TRANSACTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5010'</td>
<td>DISCARD TRANSACTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5202'</td>
<td>INQUIRE TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5204'</td>
<td>SET TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5206'</td>
<td>INQUIRE NETNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5208'</td>
<td>SET NETNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5210'</td>
<td>DISCARD TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5212'</td>
<td>INQUIRE TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5214'</td>
<td>SET TERMINAL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5216'</td>
<td>INQUIRE NETNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5402'</td>
<td>INQUIRE SYSTEM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5404'</td>
<td>SET SYSTEM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5412'</td>
<td>INQ SYSTEM</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5602'</td>
<td>SPOOLOPEN INPUT</td>
<td>API</td>
</tr>
<tr>
<td>X'5602'</td>
<td>SPOOLOPEN OUTPUT</td>
<td>API</td>
</tr>
<tr>
<td>X'5604'</td>
<td>SPOOLREAD</td>
<td>API</td>
</tr>
<tr>
<td>X'5606'</td>
<td>SPOOLWRITE</td>
<td>API</td>
</tr>
<tr>
<td>X'5610'</td>
<td>SPOOLCLOSE</td>
<td>API</td>
</tr>
<tr>
<td>X'5802'</td>
<td>INQUIRE CONNECTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5804'</td>
<td>SET CONNECTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5806'</td>
<td>PERFORM ENDAFFINITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5810'</td>
<td>DISCARD CONNECTION</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5A02'</td>
<td>INQUIRE MODENAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5A04'</td>
<td>SET MODENAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5C02'</td>
<td>INQUIRE TDQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5C04'</td>
<td>SET TDQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5C10'</td>
<td>DISCARD TDQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5E02'</td>
<td>INQUIRE TASK</td>
<td>SPI</td>
</tr>
<tr>
<td>X'5E04'</td>
<td>SET TASK</td>
<td>SPI</td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>X'5E06'</td>
<td>CHANGE TASK API</td>
<td></td>
</tr>
<tr>
<td>X'5E08'</td>
<td>INQUIRE STORAGE SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E12'</td>
<td>INQUIRE TCLASS SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E14'</td>
<td>SET TCLASS SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E18'</td>
<td>DISCARD TRANCLASS SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E1A'</td>
<td>INQUIRE TRANCLASS SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E1C'</td>
<td>SET TRANCLASS SPI</td>
<td></td>
</tr>
<tr>
<td>X'5E22'</td>
<td>WAIT EXTERNAL API</td>
<td></td>
</tr>
<tr>
<td>X'5E32'</td>
<td>WAITCICS API</td>
<td></td>
</tr>
<tr>
<td>X'6002'</td>
<td>INQUIRE JOURNALNUM SPI</td>
<td></td>
</tr>
<tr>
<td>X'6004'</td>
<td>SET JOURNALNUM SPI</td>
<td></td>
</tr>
<tr>
<td>X'6010'</td>
<td>DISCARD JOURNALNAME SPI</td>
<td></td>
</tr>
<tr>
<td>X'6012'</td>
<td>INQUIRE JOURNALNAME SPI</td>
<td></td>
</tr>
<tr>
<td>X'6014'</td>
<td>SET JOURNALNAME SPI</td>
<td></td>
</tr>
<tr>
<td>X'6202'</td>
<td>INQUIRE VOLUME SPI</td>
<td></td>
</tr>
<tr>
<td>X'6204'</td>
<td>SET VOLUME SPI</td>
<td></td>
</tr>
<tr>
<td>X'6402'</td>
<td>PERFORM SECURITY SPI</td>
<td></td>
</tr>
<tr>
<td>X'6602'</td>
<td>INQUIRE DUMPD'S SPI</td>
<td></td>
</tr>
<tr>
<td>X'6604'</td>
<td>SET DUMPD'S SPI</td>
<td></td>
</tr>
<tr>
<td>X'6612'</td>
<td>INQUIRE TRANDUMPCODE SPI</td>
<td></td>
</tr>
<tr>
<td>X'6614'</td>
<td>SET TRANDUMPCODE SPI</td>
<td></td>
</tr>
<tr>
<td>X'6622'</td>
<td>INQUIRE SYSDUMPCODE SPI</td>
<td></td>
</tr>
<tr>
<td>X'6624'</td>
<td>SET SYSDUMPCODE SPI</td>
<td></td>
</tr>
<tr>
<td>X'6802'</td>
<td>INQUIRE VTAM SPI</td>
<td></td>
</tr>
<tr>
<td>X'6804'</td>
<td>SET VTAM SPI</td>
<td></td>
</tr>
<tr>
<td>X'6812'</td>
<td>INQUIRE AUTOINSTALL SPI</td>
<td></td>
</tr>
<tr>
<td>X'6814'</td>
<td>SET AUTOINSTALL SPI</td>
<td></td>
</tr>
<tr>
<td>X'6822'</td>
<td>INQUIRE DELETISHED SPI</td>
<td></td>
</tr>
<tr>
<td>X'6824'</td>
<td>SET DELETISHED SPI</td>
<td></td>
</tr>
<tr>
<td>X'6826'</td>
<td>PERFORM DELETISHED SPI</td>
<td></td>
</tr>
<tr>
<td>X'6A02'</td>
<td>QUERY SECURITY API</td>
<td></td>
</tr>
<tr>
<td>X'6C02'</td>
<td>WRITE OPERATOR API</td>
<td></td>
</tr>
<tr>
<td>X'6C12'</td>
<td>CICSMESSAGE API</td>
<td></td>
</tr>
<tr>
<td>X'6E02'</td>
<td>INQUIRE IRC SPI</td>
<td></td>
</tr>
<tr>
<td>X'6E04'</td>
<td>SET IRC SPI</td>
<td></td>
</tr>
<tr>
<td>X'7002'</td>
<td>INQUIRE STATISTICS SPI</td>
<td></td>
</tr>
<tr>
<td>X'7004'</td>
<td>SET STATISTICS SPI</td>
<td></td>
</tr>
<tr>
<td>X'7006'</td>
<td>PERFORM STATISTICS SPI</td>
<td></td>
</tr>
<tr>
<td>X'7008'</td>
<td>COLLECT STATISTICS SPI</td>
<td></td>
</tr>
<tr>
<td>X'7012'</td>
<td>INQUIRE MONITOR SPI</td>
<td></td>
</tr>
<tr>
<td>X'7014'</td>
<td>SET MONITOR SPI</td>
<td></td>
</tr>
<tr>
<td>X'7202'</td>
<td>PERFORM RESETTIME SPI</td>
<td></td>
</tr>
<tr>
<td>X'7402'</td>
<td>SIGNON API</td>
<td></td>
</tr>
<tr>
<td>X'7404'</td>
<td>SIGNOFF API</td>
<td></td>
</tr>
<tr>
<td>X'7406'</td>
<td>VERIFY PASSWORD API</td>
<td></td>
</tr>
<tr>
<td>X'7408'</td>
<td>CHANGE PASSWORD API</td>
<td></td>
</tr>
<tr>
<td>X'7602'</td>
<td>PERFORM SHUTDOWN SPI</td>
<td></td>
</tr>
<tr>
<td>X'7802'</td>
<td>INQUIRE TRACEDEST SPI</td>
<td></td>
</tr>
<tr>
<td>X'7804'</td>
<td>SET TRACEDEST SPI</td>
<td></td>
</tr>
<tr>
<td>X'7812'</td>
<td>INQUIRE TRACEFLAG SPI</td>
<td></td>
</tr>
<tr>
<td>X'7814'</td>
<td>SET TRACEFLAG SPI</td>
<td></td>
</tr>
<tr>
<td>X'7822'</td>
<td>INQUIRE TRACETYPE SPI</td>
<td></td>
</tr>
<tr>
<td>X'7824'</td>
<td>SET TRACETYPE SPI</td>
<td></td>
</tr>
<tr>
<td>EIBFN value</td>
<td>Command</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>X'7A02'</td>
<td>INQUIRE DSNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'7A04'</td>
<td>SET DSNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'7C02'</td>
<td>INQUIRE EXCI</td>
<td>SPI</td>
</tr>
<tr>
<td>X'7E02'</td>
<td>DUMP TRANSACTION</td>
<td>API</td>
</tr>
<tr>
<td>X'7E04'</td>
<td>PERFORM DUMP</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8002'</td>
<td>INQUIRE TSQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8004'</td>
<td>SET TSQUEUE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8012'</td>
<td>INQUIRE TSONAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8014'</td>
<td>SET TSONAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'801A'</td>
<td>INQUIRE TSPOOL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8022'</td>
<td>INQUIRE TSMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8030'</td>
<td>DISCARD TSMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'820C'</td>
<td>REQUEST PASSTICKET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'820E'</td>
<td>AP</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8210'</td>
<td>ALLOCATE POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8210'</td>
<td>ALLOCATE PASSCONVID</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8212'</td>
<td>CONVERSE FORMATTED</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8214'</td>
<td>CONVERSE DATASTREAM</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8216'</td>
<td>EXTRACT CONV</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8218'</td>
<td>EXTRACT FIELD</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'821A'</td>
<td>EXTRACT STSN</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'821C'</td>
<td>FREE</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'821E'</td>
<td>ISSUE</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8220'</td>
<td>RECEIVE FORMATTED</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8222'</td>
<td>RECEIVE DATASTREAM</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8224'</td>
<td>SEND FORMATTED</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8226'</td>
<td>SEND DATASTREAM</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8228'</td>
<td>START</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'840E'</td>
<td>SP NOOP</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8422'</td>
<td>INQUIRE PROPERTYSET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8428'</td>
<td>INSTALL PROPERTYSET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8430'</td>
<td>DISCARD PROPERTYSET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8442'</td>
<td>INQUIRE NODE</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8444'</td>
<td>SET NODE</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8444'</td>
<td>SET NODELIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8448'</td>
<td>INSTALL NODELIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'844A'</td>
<td>ADD POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'844C'</td>
<td>DELETE POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8450'</td>
<td>DISCARD NODELIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8462'</td>
<td>INQUIRE POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8464'</td>
<td>SET POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8464'</td>
<td>SET POOLLIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8468'</td>
<td>INSTALL POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8470'</td>
<td>DISCARD POOL</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8482'</td>
<td>INQUIRE TARGET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8484'</td>
<td>SET TARGETLIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8484'</td>
<td>SET TARGET</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8488'</td>
<td>INSTALL TARGETLIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8490'</td>
<td>DISCARD TARGETLIST</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'84A2'</td>
<td>INQUIRE CONNECTION</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'84A4'</td>
<td>SET CONNECTION</td>
<td>FEPI</td>
</tr>
<tr>
<td>X'8602'</td>
<td>ACQUIRE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8802'</td>
<td>INQUIRE EXITPROGRAM</td>
<td>SPI</td>
</tr>
</tbody>
</table>
Table 12. EIB Field Name Values & Types (continued)

<table>
<thead>
<tr>
<th>EIBFN value</th>
<th>Command</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'8A02'</td>
<td>INQUIRE REQID</td>
<td>SPI</td>
</tr>
<tr>
<td>X'8C02'</td>
<td>WRITE MESSAGE</td>
<td>API</td>
</tr>
<tr>
<td>X'9002'</td>
<td>INQUIRE UOW</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9004'</td>
<td>SET UOW</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9022'</td>
<td>INQUIRE UOWENQ</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9042'</td>
<td>INQUIRE UOWLINK</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9044'</td>
<td>SET UOWLINK</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9062'</td>
<td>INQUIRE UOWDSNFAIL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9082'</td>
<td>INQUIRE ENQMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9084'</td>
<td>SET ENQMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9090'</td>
<td>DISCARD ENQMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9202'</td>
<td>INQUIRE JOURNALMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9210'</td>
<td>DISCARD JOURNALMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9212'</td>
<td>INQUIRE STREAMNAME</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9402'</td>
<td>INQUIRE DB2CONN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9404'</td>
<td>SET DB2CONN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9410'</td>
<td>DISCARD DB2CONN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9422'</td>
<td>INQUIRE DB2ENTRY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9424'</td>
<td>SET DB2ENTRY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9430'</td>
<td>DISCARD DB2ENTRY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9442'</td>
<td>INQUIRE DB2TRAN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9444'</td>
<td>SET DB2TRAN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9450'</td>
<td>DISCARD DB2TRAN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9602'</td>
<td>INQUIRE PROCESSTYPE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9604'</td>
<td>SET PROCESSTYPE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9610'</td>
<td>DISCARD PROCESSTYPE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9612'</td>
<td>INQUIRE ACTID</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9614'</td>
<td>INQUIRE CONTAIN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9616'</td>
<td>INQUIRE EVENT</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9618'</td>
<td>INQUIRE PROCESS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9620'</td>
<td>STARTBROWSE ACTIVITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9622'</td>
<td>GETNEXT ACTIVITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9624'</td>
<td>ENDBROWSE ACTIVITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9626'</td>
<td>STARTBROWSE CONTAINER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9628'</td>
<td>GETNEXT CONTAINER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'962A'</td>
<td>ENDBROWSE CONTAINER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'962C'</td>
<td>STARTBROWSE EVENT</td>
<td>SPI</td>
</tr>
<tr>
<td>X'962E'</td>
<td>GETNEXT EVENT</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9630'</td>
<td>ENDBROWSE EVENT</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9632'</td>
<td>STARTBROWSE PROCESS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9634'</td>
<td>GETNEXT PROCESS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9636'</td>
<td>ENDBROWSE PROCESS</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9638'</td>
<td>INQUIRE TIMER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'963A'</td>
<td>STARTBROWSE TIMER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'963C'</td>
<td>GETNEXT TIMER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'963E'</td>
<td>ENDBROWSE TIMER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9802'</td>
<td>INQUIRE CFDTPOOL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9A02'</td>
<td>INQUIRE REQUESTMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9A10'</td>
<td>DISCARD REQUESTMODEL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C02'</td>
<td>INQUIRE TCPIPSERVICE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C04'</td>
<td>SET TCPIPSERVICE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C10'</td>
<td>DISCARD TCPIPSERVICE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C12'</td>
<td>INQUIRE TCP/IP</td>
<td>SPI</td>
</tr>
</tbody>
</table>
Table 12. EIB Field Name Values & Types (continued)

<table>
<thead>
<tr>
<th>EIBFN value</th>
<th>Command</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'9C14'</td>
<td>SET TCPIP</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C22'</td>
<td>INQUIRE WEB</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9C24'</td>
<td>SET WEB</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9E02'</td>
<td>INQUIRE DOCTEMPLATE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'9E10'</td>
<td>DISCARD DOCTEMPLATE</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B002'</td>
<td>INQUIRE JVMPOOL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B004'</td>
<td>SET JVMPOOL</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B202'</td>
<td>INQUIRE CORBASERVER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B204'</td>
<td>SET CORBASERVER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B206'</td>
<td>PERFORM CORBASERVER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B210'</td>
<td>DISCARD CORBASERVER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B222'</td>
<td>INQUIRE DJAR</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B226'</td>
<td>PERFORM DJAR</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B230'</td>
<td>DISCARD DJAR</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B242'</td>
<td>INQUIRE BEAN</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B402'</td>
<td>INQUIRE BRFACILITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B404'</td>
<td>SET BRFACILITY</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B602'</td>
<td>INQUIRE DISPATCHER</td>
<td>SPI</td>
</tr>
<tr>
<td>X'B604'</td>
<td>SET DISPATCHER</td>
<td>SPI</td>
</tr>
</tbody>
</table>

For trace entries made on exit from DFHEIP, field A gives you the response code from the request. Look carefully at any response codes—they could provide the clue to the loop. Has the program been designed to deal with every possible response from DFHEIP? Could the response code you see explain the loop?

If you see a repeating pattern of trace points for AP 00E1, you have a non-yielding loop. If you can match the repeating pattern to statements in the source code for your program, you have identified the limits of the loop.

If you see no repeating pattern of trace points for AP 00E1, it is likely that you have a tight loop. The last entry for AP 00E1 (if there is one) should have been made from a point just before the program entered the loop. You might be able to recognize the point in the program where the request was made, by matching trace entries with the source code of the program.

**Using the transaction dump**

First you need to find the PSW, and see if it points into your program. This is likely to be the case if you have a tight loop, and it should lead you to an instruction within the loop.

If the next instruction address is not within your code, it is of less value for locating the loop. However, you should attempt to identify the module containing the instruction, as it is likely to be one that was called during the execution of a CICS request made within the loop. Use the module index at the end of the formatted dump to find the module name. If the PSW address is not contained in one of these areas, another program was probably executing on behalf of CICS when the runaway task timer expired.

**Note:** It is possible that the loop was contained entirely within a module owned by CICS or some other product, and your program was not responsible for it at all. If you find that the loop is contained within CICS code, contact the IBM Support Center.
If the PSW does point to a module outside your application program, you need to find the address of the return point in your program from the contents of register 14 in the appropriate register save area. The return address will lie within the loop, if the loop is not confined to system code.

When you have located a point within the loop, work through the source code and try to find the limits of the loop.

**Finding the reason for the loop**

When you have identified the limits of the loop, 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.

The nature of the symptoms may indicate which transaction is involved, but you probably need to use trace to define the limits of the loop.

Use auxiliary trace to capture the trace entries, to ensure that the entire loop is captured in the trace data. If you use internal trace, there is a danger that wraparound will prevent you from seeing the whole loop.

Use the CETR transaction to set up the following tracing options. You can use the transaction dynamically on the running CICS system. For guidance about using the CETR transaction, see Chapter 15, “Using traces in problem determination,” on page 225.

1. Select level-1 special tracing for every component, using the CETR transaction. 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.
2. Set all standard tracing off, by setting the master system trace flag off.
3. Select special tracing for just the task containing the loop.
4. Set the auxiliary tracing status to STARTED, and the auxiliary switch status to ALL. As CETR allows you to control trace dynamically, you do not need to start tracing until the task is running and the symptoms of looping appear.

These steps ensure that you get all level-1 trace points traced for just the task you suspect of looping, the trace entries being sent to the auxiliary trace destination.

When you have captured the trace data, you need to purge the looping task from the system. Use the CEMT INQ TASK command to find the number of the task, and then purge it using either the CEMT SET TASK PURGE or the CEMT SET TASK
FORCEPURGE command. This causes the transaction to abend, and to produce a transaction dump of the task storage areas.

**Note:** The use of FORCEPURGE is, in general, not recommended, because it can cause unpredictable system problems. For example, it causes task storage areas to be released, including I/O areas, without notifying any components that might be accessing them. If the FORCEPURGE task was waiting for input, such an area might be written to after it is released. The storage might even be in use by another task when the input occurs.

**The documentation you need**

In addition to the auxiliary trace data and the transaction dump, you need source listings of all the programs in the transaction.

The trace data and the program listings should enable you to identify the limits of the loop. You need the transaction dump to examine the user storage for the program. The data you find there could provide the evidence you need to explain why the loop occurred.

**Identifying the loop**

Examine the trace table, and try to detect the repeating pattern of trace entries. If you cannot do so straightaway, remember that many different programs might be involved, and the loop could be large. Another possibility is that you might not have captured the entire loop in the trace data set. This could be because the loop did not have time to complete one cycle before you purged the transaction, or the trace data sets might have wrapped before the loop was complete.

Consider also the possibility that you might not be dealing with a loop, and the symptoms you saw are due to something else—poor application design, for example.

If you are able to detect a pattern, you should be able to identify the corresponding pattern of statements in your source code.

**Note:** The PSW is of no value in locating loops that are not detected by CICS. The contents of the PSW are unpredictable, and the PSW is not formatted in the transaction dump for ATCH abends.

**Finding the reason for the loop**

Look carefully at the statements contained in the loop. Does the logic of the code suggest why the loop occurred? If not, you need to examine the contents of data fields in the task user storage. Look particularly for unexpected response codes, and null values when finite values are expected. Programs can react unpredictably when they encounter these conditions, unless they are tested for and handled accordingly.

**What to do if you cannot find the reason for a loop**

If you cannot find the reason for a non-yielding or a yielding loop using the techniques outlined above, there are two more approaches that you can adopt:

1. Use the interactive tools that CICS provides
2. Modify the program, and execute it again.
Investigating loops using interactive tools

If you have a non-yielding or a yielding loop, you can use the execution diagnostic facility (CEDF) to look at the various parts of your program and storage at each interaction with CICS. If you suspect that some unexpected return code might have caused the problem, CEDF is a convenient way of investigating the possibility.

CECI and CEBR are also useful for investigating loops. You can, for example, use them to examine the status of files and queues during the execution of your program. Programs can react unpredictably if records and queue entries are not found when these conditions are not tested for and handled accordingly.

Modifying your program to investigate the loop

If the program is extremely complex, or the data path difficult to follow, you may need to insert additional statements into the source code. Even extra ASKTIME commands allow you to use EDF and inspect the program at more points. You can also request dumps from within your program, and insert user trace entries, to help you to find the reason for the loop.
Chapter 9. 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 162.
2. Some tasks fail to get attached to the dispatcher—see “Why tasks fail to get attached to the dispatcher” on page 163.
3. Some tasks get attached to the dispatcher, but fail to get dispatched—see “Why tasks fail to get an initial dispatch” on page 165.
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 166.

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.”

There is a quick reference section at the end of this section ("A summary of performance bottlenecks, symptoms, and causes" on page 167) 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. 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)

Each bottleneck is affected by a different set of system parameters and you might find that adjusting the parameters solve the problem. It is useful to determine which bottleneck is causing your performance problem, so you can find out which parameters you need to consider.

If performance is particularly poor for any of the tasks in your system, you might be able to capture useful information about them with CEMT INQ TASK. However, tasks usually run more quickly than you can inquire on them, even though there might be a performance problem. You then need to consider using performance class monitoring or tracing to get the information you need.

Initial attach to the transaction manager

If a task has not been attached to the transaction manager, you cannot get any information about its status online. CEMT 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 162.
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 will show it to be 'SUSPENDED' on a resource type of MXT or TCLASS. These are the only valid reasons why a user task, having been attached to the transaction manager, would not be attached to the dispatcher.

If CEMT 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 "Is the MXT limit preventing tasks from getting attached?" on page 163.

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, you need to 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 165.

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 166.

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 75. 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:

- The system is at the MXT limit (see "Is the MXT limit preventing tasks from getting attached?").
- The task belongs to a transaction class that is at its MAXACTIVE limit.

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.

**Using transaction manager statistics**

To find out how often the MXT and transaction class limits are reached, look at the transaction global statistics and transaction class statistics. If you compare the number of times these limits are reached with the total number of transactions, you can see whether the values set for the limits are adversely affecting performance.

To gather statistics relating to the number of times that the MXT or transaction class limits are reached, you need to use, at the start of the run, the command CEMT PERFORM STATISTICS RECORD (or your site replacement) with the keywords TRANSACTION and TRANCLASS.

```
CEMT PERFORM STATISTICS RECORD [TRANCLASS TRANSACTION]
```

The statistics are gathered and recorded in the SMF data set. You can format this data set by using the statistics utility program, DFHSTUP. Please read the 'DFHSTUP' section in *CICS Operations and Utilities Guide* for details on how to use this facility.

When formatting the SMF data set using DFHSTUP, you may find the following DFHSTUP control parameters useful:

```
SELECT APPLID=
COLLECTION TYPE=
REQTIME START= ,STOP=
DATE START= ,STOP=
```
See the ‘DFHSTUP’ section in CICS Operations and Utilities Guide for details on how to code these parameters. If you correctly code these control parameters, you will avoid the formatting of much information that may well be unnecessary at this point.

If MXT is never reached, or reached only infrequently, it is not affecting performance.

If MXT is reached for 5% of transactions, this might have a noticeable effect on performance. When the ratio reaches 10%, there is likely to be a significant effect on performance, and this could account for some tasks taking a long time to get a first attach.

Consider revising the MXT and transaction class values if the statistics indicate that they are affecting performance. For guidance about the performance considerations when you set these limits, see CICS Performance Guide.

Using CICS monitoring
You can use monitoring information to find out how long an individual task waits to be attached to the Dispatcher. If you want to summarize the monitoring data of particular transactions to assess the impact across many tasks, you can use products such as Service Level Reporter (SLR) or Enterprise Performance Data Manager/MVS (EPDM/MVS).

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 CICS Performance Guide.

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.

If you do not want to do any other tracing, internal trace is probably a suitable destination for trace entries. Because the task you are interested in is almost inactive, very few trace entries are generated.

Select special tracing for the transaction associated with the task, and turn off all standard tracing by setting the master system trace flag off. Define as special trace points the level-1 trace points for transaction manager (XM), and for the CICS task controlling the facility that initiates the task, such as terminal control (TC). Make sure that no other trace points are defined as special. For guidance about setting up these tracing options, see Chapter 15, “Using traces in problem determination,” on page 225.

When you have selected the options, start tracing to the internal trace table and attempt to initiate the task. When the task starts, get a system dump with CEMT PERFORM SNAP. Format the dump using the keyword TR, to get the internal trace table.

Look for the trace entry showing terminal control calling the transaction manager with a request to attach the task, and the subsequent trace entry showing the
transaction manager calling dispatcher domain with a request to attach the task. The time stamps on the two trace entries tell you the time that elapsed between the two events. That is equal to the time taken for the task to be attached.

**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:

\[
\text{Delay in dispatcher} = \text{DSPDELAY} - (\text{TCLDELAY} + \text{MXTDELAY})
\]

If ‘Delay in Dispatcher’ is significantly greater than 0, the dispatcher could not dispatch the task immediately.

The factors that influence the length of time a task must wait before getting its first dispatch are:

- The priority of the task
- Whether the system is becoming short on storage.

**Priorities of tasks**

Normally, the priorities of tasks determine the order in which they are dispatched. Priorities can have any value in the range 1–255. If your task is getting a first dispatch (and, possibly, subsequent dispatches) too slowly, you might consider changing its priority to a higher value.

You have no control over the priorities of CICS system tasks.

One other factor affecting the priorities of tasks is the priority aging multiplier, PRTYAGE, that you code in the system initialization parameters. This determines the rate at which tasks in the system can have their priorities aged. Altering the value of PRTYAGE affects the rate at which tasks are dispatched, and you probably need to experiment to find the best value for your system.

Storage stress conditions are detailed in the storage manager statistics. CICS attempts to alleviate the situation by releasing programs with no current user, and by not attaching new tasks. If these actions fail to eliminate storage stress or if the SOS condition is caused by a suspended GETMAIN, one or both of these messages is sent to the console:

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

DFHSM0133 applid CICS is under stress (short on storage above 16MB)
If you don't observe the SOS messages, you can find out how many times CICS has gone SOS from the storage manager statistics ("Times went short on storage"). You can also get this information from the storage manager domain DSA summary in a formatted system dump.

**Note:** Release of the storage cushion is not the only cause of CICS going SOS. The condition is also raised if a task makes an unconditional request for storage greater than the storage cushion size when the system is approaching SOS. In such a case, the cushion is not released, but the task making the unconditional request is suspended and message DFHSM0131I or DFHSM0133I may be issued. CICS resumes the suspended tasks immediately if storage is made available by CICS releasing unused programs. The short-on-storage condition remains until all the previously suspended tasks have obtained the storage they requested.

Two other conditions are recognized by the dispatcher on the approach to SOS, namely ‘storage getting short’ and ‘storage critical’. The two conditions affect the chance of new tasks getting a first dispatch.

From the ‘storage getting short’ point, through ‘storage critical’ and right up to SOS, the priorities of new user tasks are reduced in proportion to the severity of the condition. 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 ‘storage critical’ is approached, new tasks might typically be delayed by up to a second before they are dispatched for the first time.

It is likely that ‘storage getting short’ and ‘storage critical’ occur many times for every occasion SOS is reached. If you want to see how often these points are reached, select level-2 tracing for the dispatcher domain and look out for trace point IDs **DS 0038** (‘storage getting short’) and **DS 0039** (‘storage critical’). Trace point **DS 0040** shows that storage is OK.

A summary of the effects of ‘storage getting short’, ‘storage critical’, and SOS is given in **Table 13**.

<table>
<thead>
<tr>
<th>State of storage</th>
<th>Effects on user tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage getting short</td>
<td>Priority of new user tasks reduced a little</td>
</tr>
<tr>
<td>Storage critical</td>
<td>Priority of new user tasks reduced considerably</td>
</tr>
</tbody>
</table>

### Why tasks take a long time to complete

When a ready task is dispatched, it becomes a running task. It is unlikely to complete without being suspended at least once, and it is likely to go through the ‘READY - RUNNING - SUSPENDED’ cycle several times during its lifetime in the dispatcher.

The longer the task spends in the non-running state, either ‘ready’ or ‘suspended’, the greater your perception of performance degradation. In extreme cases, the task might spend so long in the non-running state that it is apparently waiting indefinitely. It is not likely to remain ‘ready’ indefinitely without running, but it could spend so long suspended that you would probably classify the problem as a wait.
The purpose of this section is to deal not with waiting tasks, but instead with tasks that complete more slowly than they should.

Here are some factors that can affect how long tasks take to complete:

- System loading
- Time-out interval for tasks
- Distribution of data sets on DASD volumes.

Each of these factors is considered in turn.

**The effect of system loading on performance**

The most obvious factor affecting the time taken for a task to complete is system loading. For more information, see [CICS Performance Guide](#). Note in particular that there is a critical loading beyond which performance is degraded severely for only a small increase in transaction throughput.

**The effect of task time-out interval on performance**

The time-out interval is the length of time a task can wait on a resource before it is removed from the suspended state. A transaction that times out is normally abended.

Any task in the system can use resources and not allow other tasks to use them. Normally, a task with a large time-out interval is likely to hold on to resources longer than a task with a short time-out interval. Such a task has a greater chance of preventing other tasks from running. It follows that task time-out intervals should be chosen with care, to optimize the use of resources by all the tasks that need them.

**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**

<table>
<thead>
<tr>
<th>Bottleneck</th>
<th>Symptoms</th>
<th>Possible causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial attach to transaction manager</td>
<td>CEMT INQ TASK does not know task. Tracing shows long wait for attach to transaction manager.</td>
<td>• Interval on EXEC CICS START too long  • Terminal not available  • Remote system not available</td>
</tr>
</tbody>
</table>
Table 14. A summary of performance bottlenecks, symptoms and causes (continued)

<table>
<thead>
<tr>
<th>Bottleneck</th>
<th>Symptoms</th>
<th>Possible causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial attach to dispatcher</td>
<td>CEMT INQ TASK shows wait on MXT or transaction class.</td>
<td>MXT or transaction class limits set too low</td>
</tr>
<tr>
<td></td>
<td>Tracing shows long wait for attach to dispatcher.</td>
<td></td>
</tr>
<tr>
<td>First dispatch</td>
<td>Performance class monitoring shows long wait for first dispatch.</td>
<td>• MXT or transaction class limits set too low</td>
</tr>
<tr>
<td></td>
<td>Storage statistics show CICS has gone SOS.</td>
<td>• Priority of task set too low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Insufficient storage</td>
</tr>
<tr>
<td>SUSPEND / RESUME cycle</td>
<td>Tasks take a long time to complete.</td>
<td>• System loading high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Task time-out interval too large</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CICS data sets are on volumes susceptible to MVS RESERVE locking</td>
</tr>
</tbody>
</table>
Chapter 10. Dealing with incorrect output

The term “incorrect output” can be interpreted in many different ways, and its meaning for the purpose of problem determination in this section is explained in Chapter 2, “Classifying the problem,” on page 7.

The various categories of incorrect output are dealt with in:
- “Trace output is incorrect”
- “Dump output is incorrect” on page 173
- “Incorrect data is displayed on a terminal” on page 176
- “Specific types of incorrect output for terminals” on page 178
- “Incorrect data is present on a VSAM data set” on page 182
- “An application does not work as expected” on page 183
- “Your transaction produces no output at all” on page 184
- “Your transaction produces some output, but it is wrong” on page 189.

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.”
- You might have captured the wrong data. This is dealt with in “You have captured the wrong trace data” on page 170.

Tracing has gone to the wrong destination

In terms of destinations, CICS system trace entries belong to one of three groups:
- CICS trace entries, other than CICS VTAM 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 VTAM exit traces that are not exception traces go only to the GTF trace data set, if GTF tracing is active.
- CICS VTAM 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 VTAM exit traces, you can inquire on the current destinations and set them to what you want using the CETR transaction.

Figure 23 on page 247 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 master system trace flag value only determines whether standard tracing is to be done for a task (see Table 26 on page 235). 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 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 171 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 171 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” 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 25 on page 234.

You can deduce from the table that it is possible to get standard tracing when a transaction is initiated at one terminal, and special tracing when it is initiated from another terminal. This raises the possibility of setting up inappropriate task tracing options, so the trace entries that interest you—for example, when the transaction is initiated from a particular terminal—are not made.

**You are not getting the correct component tracing**
If you are not getting the correct component tracing, use the CETR transaction to inquire on the current component tracing options and, if necessary, to change them.

First, check that you are only tracing components that interest you. If some other components are being traced, change the options so they are no longer traced for standard tracing or for special tracing, as appropriate.

Next, check that the right tracing levels have been defined for standard tracing and special tracing. Remember that, whenever a task that has standard tracing is running, the trace points that you have defined as standard for a component are traced whenever that component is invoked. Similarly, special trace points are traced whenever special task tracing is being done.

Table 26 on page 235 illustrates the logic used to determine whether a trace call is to be made from a trace point.

If you are satisfied that the component tracing selectivity is correct but you are still getting too much or too little data, read “You are not getting the correct task tracing.”

**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 the immediately preceding sections You are not getting the correct task tracing and You are not getting the correct component tracing.

- 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 VTAM 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 “transaction attach” (point ID XM 1102) trace entry. This could occur if the master system trace flag is switched off and the transaction status is set to special.

  When you select trace entries by specifying TRANID or TERMID parameters in the DFHTU640 trace control statements, DFHTU640 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 DFHTU640, see the CICS Operations and Utilities Guide.

- If the options were correct and tracing was running at the right time, but the trace entries you wanted did not appear, it is likely that the task you were interested in did not run or did not invoke the CICS components you expected. Examine the trace carefully in the region in which you expected the task to appear, and attempt to find why it was not invoked. Remember also that the task tracing options might not, after all, have been appropriate.

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 bring up the CICS region in which you are interested. If you invoked the dump from the CICS master terminal using CEMT PERFORM SNAP, check that you were using the master terminal for the correct region. This is more likely to be a problem if you have a VTAM network, because that allows you to switch a single physical VTAM terminal between the different CICS 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 and Codes 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 the CICS Customization Guide.

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. For details, see Chapter 17, "Using dumps in problem determination," on page 261.

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 NOTRAN DUMP 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 15 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 15. Typical distribution of dump IDs between dump data sets

<table>
<thead>
<tr>
<th>On system dump data set</th>
<th>On transaction dump data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID=23/0001</td>
<td></td>
</tr>
<tr>
<td>ID=23/0002</td>
<td>ID=23/0002</td>
</tr>
<tr>
<td>ID=23/0004</td>
<td>ID=23/0003</td>
</tr>
<tr>
<td>ID=23/0006</td>
<td>ID=23/0005</td>
</tr>
<tr>
<td>ID=23/0007</td>
<td></td>
</tr>
<tr>
<td>ID=23/0008</td>
<td></td>
</tr>
</tbody>
</table>

For further discussion of the way CICS manages transaction and system dumps, see Chapter 17, “Using dumps in problem determination,” on page 261.

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:

- The identity of the transaction associated with the incorrect output.
- For an autoinstalled terminal, the model number, to ensure that you inquire on the correct 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 VTAM environment**

Amongst the debugging tools you have, two are likely to be of particular use for investigating terminal incorrect output errors in a VTAM environment. They are:

- VTAM buffer trace. This is a function of VTAM itself, and you need to read the appropriate manual in the VTAM library to find out how to use it.
- CICS VTAM 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 Chapter 15, "Using traces in problem determination," on page 225.
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 VTAM buffer trace to find out more about the reasons for the failure.

Unexpected messages and codes

If the “wrong data” is in the form of a message or code that you do not understand, look in the appropriate manual for an explanation of what it means.

Messages that are prefixed by DFH originate from CICS—use the CMAC transaction or look in [CICS Messages and Codes] for these. For codes that appear in the space at the bottom of the screen where status information is displayed, look in the appropriate guide for the terminal.

The following are examples of common errors that can cause messages or codes to be displayed:

- SCRNSIZE(ALTERNATE) has been specified in 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.

- EXTENDED(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 EXTENDED is NO, but check to make sure that YES has not been specified if you know your terminal is not an extended data stream device.

Unexpected appearance of uppercase or lowercase characters

If the data displayed on your terminal has unexpectedly been translated into uppercase characters, or if you have some lowercase characters when you were expecting uppercase translation, you need to look at the options governing the translation.
These are the significant properties of the various translation options you have:

- The ASIS option 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 VTAM terminals. The PROFILE UCTRAN value is valid only for VTAM terminals.
  
  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:** User exit XZCIN can also be used to perform uppercase translation.

The UPPERCASE option in the offline utilities (DFHSTUP, DFHU640, DFHTU640) specify whether all lowercase characters are to be translated to uppercase characters.

Table 16 and Table 17 summarize whether or not you get uppercase translation, depending on the values of these options.

**Table 16. Uppercase translation truth table — ASIS option not specified**

<table>
<thead>
<tr>
<th>Profile</th>
<th>TYPETERM UCTRAN(YES)</th>
<th>TYPETERM UCTRAN(NO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCTRAN(YES)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UCTRAN(NO)</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 17. Uppercase translation truth table — ASIS option is specified**

<table>
<thead>
<tr>
<th>Profile</th>
<th>TYPETERM UCTRAN(YES)</th>
<th>TYPETERM UCTRAN(NO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCTRAN(YES)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>UCTRAN(NO)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**CRTE and uppercase translation**

**Initiating a CRTE session**

The input required to start a CRTE routing session is of the form:  

```
CRTE SYSID(xxxx),TRPROF(yyyyyyyy)
```

Translation to uppercase is dictated by the typeterm of the terminal at which CRTE was entered and CRTE’s transaction profile definition as shown in Table 18 on page 180.
Table 18. Uppercase translation on CRTE session initiation

<table>
<thead>
<tr>
<th>TYPETERM</th>
<th>CRTE PROFILE</th>
<th>UCTRAN PROFILE</th>
<th>INPUT TRANSLATED TO UPPERCASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>YES/NO</td>
<td>ALL OF THE INPUT</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>NONE OF THE INPUT. See note.</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>ALL OF THE INPUT EXCEPT THE TRANSID. See note.</td>
<td></td>
</tr>
<tr>
<td>TRANID</td>
<td>YES</td>
<td>ALL OF THE INPUT</td>
<td></td>
</tr>
<tr>
<td>TRANID</td>
<td>NO</td>
<td>TRANSID ONLY</td>
<td></td>
</tr>
</tbody>
</table>

*Note:* If the transid CRTE is not entered in upper case, it will not be recognized (unless there is a lower/mixed case alias), and message DFHAC2001 will be issued.

Input within the CRTE session

During the CRTE routing session, uppercase translation is dictated by the typeterm of the terminal at which CRTE was initiated and the transaction profile definition of the transaction being initiated (which has to be a valid transaction on the application owning region) as shown in Table 19.

Table 19. Uppercase translation during CRTE session

<table>
<thead>
<tr>
<th>TYPETERM</th>
<th>TRANSACTION PROFILE (AOR)</th>
<th>UCTRAN PROFILE</th>
<th>INPUT TRANSLATED TO UPPERCASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>YES/NO</td>
<td>ALL OF THE INPUT</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>NONE OF THE INPUT. See note.</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>ALL OF THE INPUT EXCEPT THE TRANSID. See note.</td>
<td></td>
</tr>
<tr>
<td>TRANID</td>
<td>YES</td>
<td>ALL OF THE INPUT</td>
<td></td>
</tr>
<tr>
<td>TRANID</td>
<td>NO</td>
<td>TRANSID ONLY</td>
<td></td>
</tr>
</tbody>
</table>

*Note:* If the transid CRTE is not entered in upper case, it will not be recognized (unless there is a lower/mixed case alias defined on the AOR) and message DFHAC2001 will be issued.

During a CRTE routing session, if the first six characters entered at a screen are CANCEL, CICS will recognize this input in upper, lower or mixed case and end the routing session.

For more information on the ALIAS attribute of the transaction definition, see the 'Transaction' section of the [CICS Resource Definition Guide](#).

Be aware that when transaction routing from CICS Transaction Server for z/OS, Version 3 Release 1 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 may 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 VTAM terminals) that the TOR terminal uppercase translate status is copied to the AOR surrogate terminal on every flow across the link from the TOR to the AOR. Consequently:

- The EXEC CICS SET TERMINAL change of uppercase translate status will only take effect on the AOR on the next flow across the link.
- Any AOR typeterm definition used to hard code remote terminal definitions will be overridden with the TOR values for uppercase translate status.
- EXEC CICS INQUIRE TERMINAL issued on the AOR can return misleading uppercase translation status of the terminal, since the correct status on the TOR may not yet have been copied to the AOR.
- The processing of RECEIVE requests on the TOR and AOR can interrogate the uppercase translate status of the terminal. Therefore unpredictable results can also occur if the system programmer issues the EXEC CICS SET TERMINAL command during receive processing.

**CICS client virtual terminal**

If the codepage 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 containing mixed English and Katakana characters. That is because Katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as Katakana characters. If you have any Katakana terminals connected to your CICS system, specify MSGCASE=UPPER in the system initialization table to ensure that messages contain uppercase characters only.

The offline utilities DFHSTUP, DFHDU640, and DFHTU640 have an extra parameter to ensure all output is translated to uppercase. See the CICS Operations and Utilities Guide for details on how to use these parameters.

**Wrong data values are displayed**

If the data values are wrong on the user’s part of the screen (the space above the area used to display status information to the operator), or in the hard copy produced by a printer, it is likely that the application is at fault.

**Some data is not displayed**
If you find that some data is not being displayed, consider these possibilities:

- The SENDSIZE value for the 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.
- EXTENDEDDEDS(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 Application Programming Reference manual](#). Are any default values the ones you really want? Does the description of the effect of each command match your expectations?
3. Can you identify a failing sequence of commands? If so, can it be reproduced using CECI?
4. Consider the resources required by the application. Are they defined as expected?
5. Are the required functions in the failing functional area available in this system?
6. For “input” type requests, does the item exist? You can verify this using offline utilities.
7. For “output” type requests, is the item created? Verify that the before and after images are as expected.

Using traces and dumps

Traces and dumps can give you valuable information about unusual conditions that might be causing your application to work in an unexpected way.

1. If the path through the transaction is indeterminate, insert user trace entries at all the principal points.
2. If you know the point in the code where the failure occurs, insert a CICS system dump request immediately after it.
3. Use CETR to select special tracing for the level-1 trace points for all components. Select special tracing for the failing task only, and disable all standard tracing by setting the master system trace flag off.
4. Run the transaction after setting the trace options, and wait until the system dump request is executed. Format the internal trace table from the dump (formatting keyword TR), and examine the trace entries before the failure. Look in particular for unusual or unexpected conditions, possibly ones that the application is not designed to handle.
Your transaction 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.

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.

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 "No output—what to do if the task is still in the system" on page 185 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 "No output—what to do if the task is still in the system" on page 185 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 185.
No output—what to do if the task is still in the system

If you obtained no output and the task is still in the system, it is either waiting for a resource, or looping. You should get an indication of which of these two conditions is the most likely from the status for the task returned by CEMT INQ TASK.

You have a **suspended task**, treat this as a “wait” problem. Use the techniques described in Chapter 6, “Dealing with waits,” on page 49 to investigate it further.

You have a **running task**, it is likely to be looping. See Chapter 8, “Dealing with loops,” on page 143 to find out what to do next.

No output—what to do if the task is not in the system

If you have obtained no output and CEMT 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 simply that you are not setting on the START PRINTER bit in the write control character. You need to set this bit to get printed output if you have specified the STRFIELD option on a CONVERSE or SEND command, which means that the data area specified in the FROM option contains structured fields. Your application must set up the contents of the structured fields.

Your task might have been initiated by direct request from a terminal, or by automatic task initiation (ATI). Most of the techniques apply to both sorts of task, but there are some extra things to investigate for ATI tasks. Carry out the tests which apply to all tasks first, then go on to the tests for ATI tasks if you need to.

Did the task run? Techniques for all tasks

There are many different techniques for finding out if a transaction started, or if it ran but produced no output. Use the ones that are most convenient at your installation.

**Using CICS system trace entry points:**

CICS system tracing is probably the most powerful technique for finding out whether a transaction ever started. You might need to direct the trace output to the auxiliary trace destination, depending on how certain you can be about the time the task is expected to start. Even a large internal trace table might wrap and overlay the data you want to see if you are not too sure about when the task should start.

You need to use the CETR transaction to set up the right tracing options. See Chapter 15, “Using traces in problem determination,” on page 225 for guidance about setting up trace options.

Select special tracing for just your task, and disable tracing for all other tasks by setting the master system trace flag off. Set up special tracing for the level one trace points for the components that are likely to be used during the invocation of the task. The components you choose will depend on how the task is initiated—by direct request from a terminal, or by automatic transaction initialization—but they should include loader domain (LD), program manager (PG), transaction manager.
(XM), and dispatcher domain (DS). Make sure that special tracing is disabled for all other components, to minimize the amount of trace data that is collected and the tracing overhead.

Now turn tracing on, and attempt to start your task. When you are sure that the time has passed when the output should have appeared, stop tracing, and format the trace data set.

If your transaction ran, you should see the following types of trace entries for your task and the programs associated with it:
1. Loader domain, when it loaded your program, if the program was not already in main storage.
2. Transaction manager, when it attached your task to the dispatcher.
3. Dispatcher domain, when your task got its first dispatch. You might also see subsequent entries showing your task being suspended, and then resumed.
4. Program manager, for any program management functions associated with your task.

If trace entries for any of these processes are missing, that should help you to find where the failure occurred.

**Using EDF**
If the transaction being tested requires a terminal, you can use EDF. You need two other terminals for input, as well as the one that the transaction requires ("tttt"). Use one of these others to put the transaction terminal under control of EDF, with:

CEDF tttt

At the remaining terminal, enter whatever transaction or sequence of transactions causes the one under test to be initiated. Wait long enough for it to start. If no output appears at the second terminal, the transaction has not started. If you have not yet done so, consider using trace to get more information about the failure.

**Using CEDX**
You can use CEDX to debug non-terminal 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.

**Using statistics**
If no one else is using the transaction in question, you can tell from CICS statistics whether the program has been executed or not.

Use the command CEMT PERFORM STATISTICS RECORD (or your site replacement) before you test your transaction, using the TRANSACTION option:

CEMT PERFORM STATISTICS RECORD
[TRANSACTION]

This causes statistics on transactions that have been executed to be recorded in the SMF data set.

Now initiate the transaction and wait until it should have been executed. Repeat the CEMT PERFORM STATISTICS RECORD command, to get a new set of statistics written to the SMF data set. Format the data from the SMF data set for the APPLID
that interests you, and look at the statistics recorded before and after you attempted to execute the transaction. If the count for your transaction increased by 1, it was executed. If it remained the same, it was not executed.

Alternatively, if no one else is using the transaction, you can tell, using CEMT, whether the program is being executed. Use the command CEMT INQUIRE PROGRAM(xxxxxxxx) where xxxxxxxx is the program name. The screen presented to you includes a USECOUNT value. This value is the number of times that the program has been executed since the start of the current CICS session.

Now initiate the transaction and wait until it should have been executed. Repeat the CEMT INQUIRE PROGRAM(xxxxxxxx) and the USECOUNT value will have been incremented if the program has been executed.

**Formatting the SMF data set.** The statistics utility program, DFHSTUP, prepares and prints reports offline using the data recorded in the SMF data set. Details on how to use this utility program are described in the *CICS Operations and Utilities Guide*.

When you format the SMF data set using DFHSTUP in order to look at the statistics relating to executed transactions and programs, you may find the following DFHSTUP control parameters useful:

```
SELECT APPLID=
COLLECTION TYPE=REQ
TIME START= ,STOP=
DATE START= ,STOP=
```

See the ‘DFHSTUP’ section in the *CICS Operations and Utilities Guide* for details on how to code these parameters. If you correctly code these control parameters, you avoid the formatting of much information that might be unnecessary at this point.

**Using CEBR**

You can use CEBR to investigate your transaction if the transaction reads or writes to a transient data queue, or writes to a temporary storage queue. A change in such a queue is strong evidence that the transaction ran, provided that the environment is sufficiently controlled that nothing else could produce the same effect. You need to be sure that no other transaction that might be executed while you are doing your testing does the same thing.

The absence of such a change does not mean that the transaction did not run—it may have run incorrectly, so that the expected change was not made.

**Using CECI**

If your transaction writes to a file, you can use CECI before and after the transaction to look for evidence of the execution of your transaction. A change in the file means the transaction ran. If no change occurred, that does not necessarily mean that the transaction failed to run—it could have worked incorrectly, so that the changes you were expecting were not made.

**Disabling the transaction**

If your transaction requires a terminal, you can do the following. Use CEMT to disable the transaction under test, then do whatever causes the transaction to be initiated. You should get this message at the terminal where it is due to run:

```
DFHAC2008 date time applid Transaction tranid has been disabled and cannot be used
```
If you do not get this message, it is likely that your transaction did not start because of a problem with that terminal.

**Investigating tasks initiated by ATI**

In addition to the general techniques for all tasks described above, there are some additional ones for tasks that should have started by ATI.

Tasks to be started by ATI can be invoked in any of these ways:
- By issuing EXEC CICS START commands, even if no interval is specified
- By BMS ROUTE operations
- By writing to transient data queues with nonzero trigger levels.

There are many reasons why automatically initiated tasks could fail to start. Even when the CICS system is operating normally, an ATI transaction might fail to start for any of the following reasons:
- It might require a resource that is not available. The resource is usually a terminal, although it could be a queue.
- It might not be scheduled to start until some time in the future. START commands and output sent with BMS ROUTE are both subject to this sort of scheduling, but transactions started when transient data trigger levels are reached are not.

CICS maintains two chains for scheduling transactions that have been requested, but not started. They are the interval control element (ICE) chain, and the automatic initiate descriptor (AID) chain. The information contained in one or other of the chains can sometimes indicate why your task has failed to start.

**The ICE chain**

The ICE chain is used for tasks scheduled to start after some specified interval, for example on an EXEC CICS START command. You can locate it in the formatted system dump by looking at the ICP section. Look in field ICETRNID of each ICE (the 4-character transaction ID) to see if it relates to your task.

If you find an ICE for your task, look in field ICEXTOD. That will show you the expiration time of day. Does it contain the value you expect? If not, either the task which caused this one to be autoinitiated was in error, or there is a system problem.

**The AID chain**

The AID chain is used for tasks that are due to start immediately. Tasks are moved from the ICE chain to the AID chain as soon as the scheduled time expires, and they are placed there directly if there is no time delay requested. If a task needs a resource, usually a terminal, that is unavailable, the task remains on the AID chain until it can use the resource.

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')**

<table>
<thead>
<tr>
<th>Content</th>
<th>Offset</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDBMS</td>
<td>X'80'</td>
<td>BMS AID</td>
</tr>
<tr>
<td>AIDPUT</td>
<td>X'50'</td>
<td>Start with data</td>
</tr>
<tr>
<td>AIDINT</td>
<td>X'40'</td>
<td>Start with no data</td>
</tr>
<tr>
<td>AIDTDP</td>
<td>X'10'</td>
<td>Transient data AID</td>
</tr>
</tbody>
</table>
### AIDSTATI (X’2E’)  
AID status indicator:

<table>
<thead>
<tr>
<th>Content</th>
<th>Offset</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDPRIV</td>
<td>X’80’</td>
<td>Privileged allocate</td>
</tr>
<tr>
<td>AIDSENT</td>
<td>X’40’</td>
<td>This has been sent to the TOR by CRSR</td>
</tr>
<tr>
<td>AIDCANCNL</td>
<td>X’20’</td>
<td>Cancel this AID</td>
</tr>
<tr>
<td>AIDROUTP</td>
<td>X’10’</td>
<td>Not yet routed to the AOR</td>
</tr>
<tr>
<td>AIDSHIPD</td>
<td>X’08’</td>
<td>Prevent duplicate send</td>
</tr>
<tr>
<td>AIDREMX</td>
<td>X’04’</td>
<td>AID for a remote transaction</td>
</tr>
<tr>
<td>AIDREMT</td>
<td>X’02’</td>
<td>AID for a remote terminal</td>
</tr>
<tr>
<td>AIDSTTSK</td>
<td>X’01’</td>
<td>Task already initiated</td>
</tr>
</tbody>
</table>

### 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 184. 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 if 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. It is often possible for such a
person to see weaknesses in the code which you have overlooked.

Interactive tools allow you to look at the ways in which the data values being
manipulated by your program change as the transaction proceeds.

- CEDF is, perhaps, the most powerful interactive tool for checking your
  programming logic. You can use it to follow the internal flow from one CICS
  command-level statement to another. If necessary, you can add CICS statements
  such as ASKTIME at critical points in your program, to see if certain paths are
  taken, and to check program storage values.
CECI allows you to simulate CICS command statements. Try to make your test environment match the environment in which the error occurred as closely as possible. If you do not, you might find that your program works with CECI, but not otherwise.

CEBR enables you to look at temporary storage and transient data queues, and to put data into them. This can be useful when many different programs use the queues to pass data.

Note: When you use CEBR to look at a transient data queue, the records you retrieve are removed from the queue before they are displayed to you. This could alter the flow of control in the program you are testing. You can, however, use CEBR to copy transient data queues to and from temporary storage, as a way of preserving the queues if you need to.

User tracing allows you to trace the flow of control and data through your program, and to record data values at specific points in the execution of the transaction. You could, for example, look at the values of counters, flags, and key variables during the execution of your program. You can include up to 4000 bytes of data on any trace entry, 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 CICS Application Programming Reference.

CSFE storage freeze can be used to freeze the storage associated with a terminal or a transaction so that it is not FREEMAINed at the end of processing. This can be a useful tool if, for example, you want to investigate possible storage violations. You need to get a transaction dump to look at the storage after you have run the task with storage freeze on.

For long-running tasks, there is a possibility that a large amount of storage may be consumed because it cannot be FREEMAINed while storage freeze is on. For short-running tasks, however, there should be no significant overhead.

If, after using these techniques, you can find no fault with the logic of the program, the fault either lies with the way data is mapped to the terminal, or you could have missed some important evidence.

Is the data being mapped incorrectly to the terminal?

Incorrect data mapping to a terminal can have both application-related and system-related causes. If you are using BMS mapping, check the items below.

- Examine the symbolic map very carefully to make sure that it agrees with the map in the load module. Check the date and time stamps, and the size of the map.
- Make sure that the attributes of the fields are what they should be. For example:
  - An attribute of DARK on a field can prevent the data in the field from being displayed on the screen.
  - Failing to turn on the modified data tag (MDT) in a field might prevent that field from being transmitted when the screen is read in.

Note: The MDT is turned on automatically if the operator types data in the field. If, however, the operator does not type data there, the application must turn the tag on explicitly if the field is to be read in.
• If your program changes a field attribute byte, or a write control character, look at each bit and check that its value is correct by looking in the appropriate reference manual for the terminal.
Chapter 11. Dealing with storage violations

The following topics describe how to deal with storage violations:

- “Avoiding storage violations”
- “Two kinds of storage violation” on page 194
- “CICS has detected a storage violation” on page 194
- “Storage violations that affect innocent transactions” on page 199
- “Programming errors that can cause storage violations” on page 200
- “Storage recovery” on page 201

Avoiding storage violations

CICS provides three facilities that help to prevent storage violations.

**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 the CICS Customization Guide.

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 X'code') has been detected by module modname.

If you have received this message, turn first to the description of message DFHSM0102 in [CICS Messages and Codes](#) to see an explanation of the message, and then to [CICS 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."

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 199.

CICS has detected a storage violation

CICS can detect storage violations when:

1. The duplicate storage accounting area (SAA) or the initial SAA of a TIOA storage element has become corrupted.
2. The leading storage check zone or the trailing storage check zone of a user-task storage element has become corrupted.

CICS detects storage violations involving TIOAs by checking the SAA chains when it receives a command to FREEMAIN an individual element of TIOA storage, at least as far as the target element. It also checks the chains when it FREEMAINs the storage belonging to a TCTTE after the last output has taken place. CICS detects storage violations involving user-task storage by checking the storage check zones of an element of user-task storage when it receives a command to FREEMAIN that element of storage. It also checks the chains when it FREEMAINs all the storage belonging to a task when the task ends.

The storage violation is detected not at the time it occurs, but only when the SAA chain or the storage check zones are checked. This is illustrated in [Figure 18 on page 195](#) 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 18.

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 199.

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.

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 201.
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. Otherwise, go on to What the CICS system dump can tell you.

What the transaction abend message can tell you

If you get a transaction abend message, it is very likely that CICS detected the storage violation when it was attempting to satisfy a FREEMAIN request for user storage. Make a note of the information the message contains, including:

- The transaction abend code.
- The identity of the transaction whose storage has been violated.
- The identity of the program running at the time the violation was detected.
- The identity of the terminal at which the task was started.

Because CICS does not detect the overlay at the time it occurs, the program identified in the abend message 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 Chapter 15, “Using traces in problem determination,” on page 225.)

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 20 on page 197 shows the CSFE DEBUG options and their effects. Table 21 shows the startup overrides that have the same effects.

Table 20. Effects of the CSFE DEBUG transaction

<table>
<thead>
<tr>
<th>CSFE syntax</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSFE DEBUG, CHKSTSK=CURRENT</td>
<td>This checks storage check zones for all storage areas on the transaction storage chain for the current task only.</td>
</tr>
<tr>
<td></td>
<td>If a task is overlaying one of the storage check zones of its own user storage, use CSFE DEBUG,CHKSTSK=CURRENT</td>
</tr>
<tr>
<td>CSFE DEBUG, CHKSTRM=CURRENT</td>
<td>This checks SAAs for all TIOAs linked off the current TCTTE. Use this if the SAA of a TIOA has been overlaid.</td>
</tr>
<tr>
<td>CSFE DEBUG, CHKSTSK=NONE</td>
<td>This turns off storage zone checking for transaction storage areas.</td>
</tr>
<tr>
<td>CSFE DEBUG, CHKSTRM=NONE</td>
<td>This turns off SAA checking for TIOAs.</td>
</tr>
</tbody>
</table>

Table 21. Effects of the CHKSTSK and CHKSTRM overrides

<table>
<thead>
<tr>
<th>Override</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKSTSK=CURRENT</td>
<td>As CSFE DEBUG,CHKSTSK=CURRENT</td>
</tr>
<tr>
<td>CHKSTRM=CURRENT</td>
<td>As CSFE DEBUG,CHKSTRM=CURRENT</td>
</tr>
<tr>
<td>CHKSTSK=NONE</td>
<td>As CSFE DEBUG,CHKSTSK=NONE. This override is the default.</td>
</tr>
<tr>
<td>CHKSTRM=NONE</td>
<td>As CSFE DEBUG,CHKSTRM=NONE. This override is the default.</td>
</tr>
</tbody>
</table>
Your strategy should be to have the minimum tracing that will capture the storage violation, to reduce the processing overhead and to give you less trace data to process. Even so, you are likely to get a large volume of trace data, so direct the tracing to the auxiliary trace data sets. For general guidance about using tracing in CICS problem determination, see Chapter 15, “Using traces in problem determination,” on page 225.

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 25 on page 234.)

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 CICS 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 200.

Storage violations that affect innocent transactions

Storage violations that affect innocent transactions—that is, transactions that do not cause the violation—usually go undetected by CICS. However, occasionally CICS detects that the initial SAA of a TIOA element or the storage check zone of a user-storage element has been overlaid by a task that does not own it.

If they are reproducible, storage violations of this type typically occur at specific offsets within structures. For example, the start of an overlay might always be at offset 30 from the start of a field.

The most likely cause of such a violation is a transaction writing data to a part of the DSAs that it does not own, or possibly FREEMAINing such an area. The transaction might previously have GETMAINed the area and then FREEMAINed it before writing the data, or addressability might otherwise not have been correctly maintained by an application. Another possible reason is that an ECB might have been posted by a transaction after the task that was waiting on it had been canceled.

Storage violations affecting innocent transactions are, in general, more difficult to resolve than those that are detected by CICS. Often, you become aware of them long after they occur, and then you need a long history of system activity to find out what caused them.

A strategy for storage violations affecting innocent transactions

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

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 master system trace flag off.

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.

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.

The next job is to 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.

When you have found a likely trace entry, possibly showing a GETMAIN or FREEMAIN addressing the area, you need to 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.

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 may suggest coding a global trap/trace exit to detect the storage violation.

Programming errors that can cause storage violations

The purpose of this section is to outline a number of commonly occurring programming errors that can cause storage violations.

1. Failing to GETMAIN sufficient storage. This is often caused by failure to recompile all the programs for a transaction after a common storage area has been redefined with a changed length.
2. Runaway subscript. Make sure that your tables can only grow to a finite size.
3. Writing data to an area after it has been FREEMAINed.
   When a task FREEMAINs an area that it has been addressing, it can no longer write data to the area without the risk of overwriting some other data that might subsequently be there.
4. Hand posting an ECB for a canceled task.
If a task waiting on a CICS ECB is canceled, and then a transaction attempts to hand post the ECB when the resource being waited on becomes available, it may corrupt data belonging to some unrelated activity if the area once occupied by the ECB has been reused.

Storage recovery

The STGRCVY system initialization parameter enables you to vary the action taken by CICS on detection of a storage violation.

In normal operation, CICS sets up four task-lifetime storage subpools for each task. Each element in the subpool starts and ends with a check zone that includes the subpool name. At each FREEMAIN, and at end of task, CICS inspects the check zones and abends the task if either has been overwritten.

Terminal input-output areas (TIOAs) have similar check zones, each of which is set up with the same value. At each FREEMAIN of a TIOA, CICS inspects the check zones and abends the task if they are not identical.

If CICS is initialized with STGRCVY(YES), the overwriting of check zones is treated differently. After the system dump has been taken for the storage violation, CICS resets the check zones to their initial value and the task continues execution.

STGRCVY(NO) is the default.
Chapter 12. Dealing with external CICS interface (EXCI) problems

For detailed problem determination information about the external CICS interface including information about trace, system dumps and MVS abends, see the CICS External Interfaces Guide.

The following CICS messages support the external CICS interface:

DFHIR3799
DFHEX0001 DFHEX0011
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 and Codes.

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 and/or an MVS GTF data set. They are listed in CICS 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.
Chapter 13. Dealing with MRO problems

Suppose an error is suspected in communication between System A and System B. The problem can be determined by looking at either system. The following procedure applies to System A, but could equally well apply to System B. The first step is to look at the field CSACRBA in the CSA optional features list in System A. If CSACRBA is zero, the interregion communication routine is not active in that system.

If CSACRBA is not zero, examine the TCTSEs in System A and find the TCTSE for System B. In this TCTSE, TCSEIRCF is the flag byte that indicates the state of communication between the two systems. If bit TCSEIRNC in this byte is on, there is no communication between System A and System B. This is either because System B has not started interregion services, or because System A is out of service with respect to System B, or because System B is out of service with respect to System A.

If bit TCSEIRNC is off, a session should exist. The next step is to inspect the primary and secondary session(s) between the systems. The first primary session is pointed to by field TCSEVC1 in the TCTSE, and the first secondary session is pointed to by TCSEVC2. If System A initiated the session, look at secondary sessions, otherwise look at primary sessions.

Each session is defined by a TCTTE. The field TCTESCCB in TCTTE is zero if the session is not connected to the other system, otherwise it contains the address of the subsystem connection control block (SCCB) that the interregion SVC routine uses to represent that end of the connection.

Assuming the session is connected, TCTTECA is zero if no task is using the session. Otherwise, TCTTECA points to the TCA of the task that uses the session.

The protocol for interregion SVC transfer is similar to that for VTAM SNA data flow control. Field TCTEIRF1 contains information on the state of the session, field TCTESBRS gives the bracket status, field TCTESRHI is the inbound request header, and field TCTESRHO is the outbound request header.

The field TCTENIBA points to the TCTTE extension for the NIB descriptor. Within this TCTTE extension, TCTEPSQ contains the primary name, and TCTESSQ contains the secondary name. Thus a session in System A can be related to a session in System B.
Chapter 14. 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
- User journals.

This Chapter contains the following topics:

- "Categories of problem"
- "Exceeding the capacity of a log stream" on page 208
- "How CICS checks for the availability of the MVS logger" on page 208
- "Some conditions that cause CICS log manager error messages" on page 209
- "Diagnosing problems in the MVS logger" on page 213.

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 may 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 221.

   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 the CICS Transaction Server for z/OS Installation Guide 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 the CICS Operations and Utilities Guide:

- 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(ddd), where ddd 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:

- If you are using OS/390® Release 2.10 or z/OS Release 1.1, and the MVS logger PTF UW79565 has been applied, CICS checks by querying the system log connection status. This is the more efficient of the two procedures.
- If your operating system does not meet these conditions, CICS checks by initiating a BROWSE START operation on the system log.

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, as follows:

- 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.

The statistics field IGXQUERY in the CICS log manager statistics enables you 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 213.

**Note:** For details of all the return and reason codes for the MVS logger macros, see **z/OS MVS Programming: Authorized Assembler Services Reference ENF-IxG**.
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 IXGDELETE 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 219 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 213 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 213 for details.

**Loss of log data with DFHLG0740**

If DFHLG0772 gives the MVS system logger reason code as `IxgRsnCodeLoss0fDataGap (00000848)`, 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 (`IxgRsnCodeLoss0fDataGap`), 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-flight tasks 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 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. 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 the CICS Operations and Utilities Guide. For more details of a diagnostic run, see "The CICS diagnostic run mechanism" on page 221.

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. 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. Look at the following:

- "Console messages and dumps" on page 214
- "GRS resource contention" on page 214
- "Checking coupling facility structure and couple data set status" on page 215
- "Checking log stream status" on page 216
- "SMF and RMF statistics" on page 218
- "Obtaining MVS logger and coupling facility dumps" on page 218
- "Restarting the MVS logger address space" on page 221

Chapter 14. Dealing with log manager problems 213
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 the OS/390 MVS Assembler Services Reference manual.

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 R0 *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/SR 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/SR OWN/WAIT
MV26 MSDLCEIC 002F 008F6370 EXCLUSIVE OWN
S=STEP SYSZLOGR 93
SYSNAME JOBNAME ASID TCBADDR EXC/SR OWN/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 JOBNAM 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 JOBNAM 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 JOBNAM ASID TCBADDR EXC/SHR OWN/WAIT
MV29 PAYROL91 0042 007D9108 SHARE OWN
MV29 PAYROL93 0044 007D9138 SHARE OWN
S=SYSTEMS DFHSTRM PAYROL.IYK8ZET1.DFHLOG
SYSNAME JOBNAM ASID TCBADDR EXC/SHR OWN/WAIT
MV29 PAYROL91 0042 007D9108 EXCLUSIVE OWN
MV29 PAYROL91 0042 007D9108 EXCLUSIVE OWN
S=SYSTEMS DFHSTRM PAYROL.IYK8ZET3.DFHSHUNT
SYSNAME JOBNAM ASID TCBADDR EXC/SHR OWN/WAIT
MV29 PAYROL91 0042 007D9108 EXCLUSIVE OWN
MV29 PAYROL93 0044 007D9138 EXCLUSIVE OWN
S=SYSTEMS DFHSTRM PAYROL.IYK8ZET3.DFHJ02
SYSNAME JOBNAM ASID TCBADDR EXC/SHR OWN/WAIT
MV29 PAYROL91 0042 007D9108 SHARE OWN
MV29 PAYROL93 0044 007D9138 SHARE OWN
S=SYSTEMS DFHSTRM PAYROL.IYK8ZET3.DFHLOG
SYSNAME JOBNAM 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:

```
IXC358I 14.47.51 DISPLAY XCF 391
LOGR COUPLE DATA SETS
```
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:

```plaintext
/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 19 shows a typical response to this command, with system logs streams for CICS region IYLX4.

```
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:

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000

LOG STREAM CONNECTION INFO:

SYSTEMS CONNECTED: 1

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>STRUCTURE</th>
<th>CON CONNECTION</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV28</td>
<td>0000000000000000</td>
<td>00</td>
<td>00000000</td>
</tr>
</tbody>
</table>

LOG STREAM DATA SET INFO:

DATA SET NAMES IN USE: IXGLOGR.WILLIN.IYLX4.DFHLOG.<SEQ#>

Ext. <SEQ#> Lowest Blockid Highest GMT Highest Local

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*00001</td>
<td>00000007</td>
<td>00000000000496BA</td>
<td>07/18/97 08:29:13 07/18/97 09:29:</td>
</tr>
</tbody>
</table>

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.A00000404

NUMBER OF POSSIBLE ORPHANED LOG STREAM DATA SETS: 2
```

Figure 19. Example output produced by the LIST LOGSTREAM NAME command (Part 1 of 2)
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 the OS/390
MVS Setting Up a Sysplex manual.

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,
may 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, RGB, SQA, TRT, CSA, GR, 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 the commands are issued. However, it is usually more useful to take a dump at the time 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 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 20 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 log stream.

```
SLIP SET, IF, LPAMOD=(IGC0003E, 0), DATA=(1R?+4,EQ,C4C6C03,+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=structure, LOCKENTRIES, ACC=NOLIM, <change STRNAME
LISTNUM=ALL),

-->response yy

yy, ENTRYDATA=SERIALIZE, ADJUNCT=CAPTURE), S DATA=(RGB, XESDATA, ALLNUC, CS, LSQA, PSA, SQA, SWA, TRT, COUPLE, WLM, GR, XESDATA),

-->response zz

zz, ID=LOGR, REMOTE=(JOBLIST, DSPNAME, SDATA), END
```

Figure 20. An example SLIP trap. The trap is triggered when CICS issues a DFHLG0772 message. It captures dumps of the CICS address space, the MVS logger address space, and the coupling facility structure associated with the failing log stream.
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)
```

DFHLG0772 <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 DFHLG07xx range, alter the value of the EQ parameter to:

```
+4,EQ,C4C6C8D3,+8,EQ,C7F0F7),
```

DFHLG07 <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:

```
//LOGRPT EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*  
//SYSIN DD * 
DATA 
    TYPE(LOGR) 
    REPORT(YES) 
    LIST LOGSTREAM NAME(logstream_name) DETAIL(YES)
```

Figure 19 on page 217 shows example output produced by the LIST LOGSTREAM NAME command. Search for the log stream name; the structure name follows it.

Step 3 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).

Notes:

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 the OS/390 MVS Diagnostics: Tools and Service Aids manual, SY28-1085-03.
Restarting the MVS logger address space

If the MVS logger address space has failed, it can be restarted by the command `S IXGLOGRS`.

Note the ‘S’ at the end—IXGLOGRS restarts IXGLOGR as a system address space.

After the MVS logger has restarted, you must restart all CICS regions.

**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. 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.

The CICS diagnostic run mechanism

One way to gather the required diagnosis information is to 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. To supplement DFHJUP’s output, you should perform a **diagnostic run** of CICS, using the corrupt system log, before performing the initial start. On a diagnostic run, CICS:

1. Produces a dump of the CICS system state, retrieved from the failed system log.
2. 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 222.
How to perform a diagnostic run

For a diagnostic run to take place, AUTO must be specified 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.

There may be occasions other than when the system log has become unusable when you feel it would be useful to perform a diagnostic run. You can use the recovery manager utility program, DFHRMUTL, to specify a diagnostic run. For information about DFHRMUTL, see the CICS Operations and Utilities Guide.

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. 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 219. 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:

1. Set the trap for the specific DFHLG07xx message that CICS issued when the original failure occurred—see the example SLIP on page 219. When the diagnostic run occurs and the failure repeats, the message will drive the SLIP.
   
   Occasionally, the DFHLG07xx message that was issued at the time of the original failure is not repeated during a diagnostic run—instead, a different DFHLG07xx message is issued. Therefore the SLIP is not triggered. If this happens, perform another diagnostic run. This time, however, set the SLIP for the DFHLG07xx message that was issued during the first diagnostic run.

2. You do not need to specify a dump of the CICS system, because one is taken automatically by the diagnostic run mechanism. Change the JOBLIST parameter in the example SLIP to read:

   JOBLIST=(IXGLOGR,XCFAS),

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 (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.

   If DFHLOG and DFHSHUNT use separate coupling facility structures, dump both structures. Specify the names of both structures on the STRLIST parameter.
Part 3. Using traces and dumps in problem determination

This part contains:

- Chapter 15, "Using traces in problem determination," on page 225
- Chapter 17, "Using dumps in problem determination," on page 261
- Chapter 19, "The global trap/trace exit," on page 305
Chapter 15. Using traces in problem determination

This chapter describes the types of general and specialized trace available to you for problem determination, how to control the amount of tracing that CICS produces, and how to format and interpret tracing.

The types of tracing that can be used for CICS systems are:

- CICS tracing, which is performed by the trace domain at predetermined trace points in CICS code during the regular flow of control. This includes user tracing from applications. You get this when you turn on CICS internal tracing, auxiliary tracing, and GTF tracing. You control this type of tracing to suit your needs, except that, when an exception condition is detected by CICS, it always makes an exception trace entry. You cannot turn exception tracing off.
- CICS exit programming interface (XPI) tracing, which uses the TRACE_PUT XPI call from an exit program. You can control this within the exit program, or by enabling and disabling exits.
- CICS XRF tracing, which records CICS XRF-related activities. This is always running if you are operating in a CICS XRF environment.
- Program check and abend tracing, which is used by CICS to record pertinent information when a program check or abend occurs. This is controlled by CICS code.
- CICS VTAM exit tracing. The exits are driven by VTAM when it reaches a particular stage in its asynchronous processing, but the trace points are in CICS code. You can turn CICS VTAM exit tracing on or off.
- VTAM buffer tracing. This is a part of VTAM, but it can be used to record the flow of data between logical units in the CICS environment. You can control this type of tracing to meet your needs.

In addition to the general trace produced by CICS, there are a number of other, more specialized forms of trace that you can use. These are described as follows:

- “CICS exception tracing” on page 228
- “CICS XRF tracing” on page 229
- “Program check and abend tracing” on page 231
- “CICS VTAM exit tracing” on page 231
- FEPI trace.

For information about using trace to solve FEPI problems, see FEPI trace.

You have a large amount of control over the amount of CICS tracing that is done. There are a number of selection mechanisms available to you to control the extent of CICS tracing carried out in the system. These are:

- “Selecting tracing by transaction” on page 233
- “Selecting tracing by component” on page 235
- “Selecting trace destinations and related options” on page 244
- “Setting the tracing status” on page 247

You can select any combination of internal tracing, auxiliary tracing and GTF tracing to be active at the same time. Your choice has no effect on the selectivity with which system tracing is done, but each type of tracing has a set of characteristic properties. These properties are described in “CICS internal trace” on page 245, “CICS auxiliary trace” on page 245, and “CICS GTF trace” on page 246.
CICS tracing

General CICS tracing is handled by trace domain. It traces the flow of execution through CICS code, and through your applications as well. For programming information about how to make trace calls from within your own programs, see the CICS Application Programming Reference. 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.

Trace points

Trace points are included at specific points in CICS code; from these points, trace entries can be written to any currently selected trace destination. All CICS trace points are listed in alphanumeric sequence in CICS Trace Entries.

Trace levels

Some trace points are used to make exception traces when exception conditions occur, and some are used to trace the mainline execution of CICS code. Trace points of the latter type each have an associated "level" attribute. The value of this attribute depends on where the trace point is, and the sort of detail it can provide on a trace call.

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 master 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.

Trace levels can, in principle, vary in value in the range 1–32, but in practice nearly all mainline trace points have a trace level of 1 or 2.
**Level-1 trace points** are designed to give you enough diagnostic information to fix "user" errors. The following is a summary of where they are located, and a description of the information they return:

- On entry to, and exit from, every CICS domain. The information includes the domain call parameter list, and data whose address is contained in the parameter list if it is necessary for a high-level understanding of the function to be performed.
- On entry to, and exit from, major internal domain functions. The information includes parameters passed on the call, and any output from the function.
- Before and after calls to other programs, for example, VTAM. The information includes what request is to be made, the input parameters on the request, and the result of the call.
- At many of the points where trace calls were made in CICS/MVS Version 2. The type of information is the same as for that release.

**Level-2 trace points** are situated between the level-1 trace points, and they provide information that is likely to be more useful for fixing errors within CICS code. You probably will not want to use level-2 trace points yourself, unless you are requested to do so by IBM support staff after you have referred a problem to them.

**Level-3 trace points** and above are reserved for special cases. Very few components have trace points higher than 2, and they are only likely to be of use by IBM support staff. The SJ domain uses trace levels 29–32 to control JVM tracing, but these correspond to JVM trace levels 0, 1, and 2, plus a user-definable trace level.

You can select how much CICS system tracing is to be done on the basis of the trace level attributes of trace points. You can make your selection independently for each CICS component, and you can also vary the amount of tracing to be done for each task. This gives you control over what system tracing is done.

**Note:** In the storage manager component (SM), two levels of tracing, level 3 and level 4, are intended for IBM field engineering staff. These trace levels take effect only if specified in system initialization parameters and modify the internal SM operation for CICS subpools as follows:

**SM level 3 trace**

The quickcell mechanism is deactivated. Every CICS subpool, regardless of quickcelling requirements, will issue domain calls for getmain and freemain services, and these calls will be traced.

**SM level 4 trace**

Subpool element chaining on every CICS subpool is forced. Every CICS subpool, regardless of element chaining requirements, will use element chaining.

A significant performance overhead is introduced into your CICS system if these storage manager trace levels are selected. Note that specifying SM=ALL activates SM trace levels 1, 2, 3, and 4.
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 master system trace flag off.
3. Set the auxiliary trace status to STARTED, and the auxiliary trace data set and the auxiliary switch status to whatever values you want.

Exception traces are now made to an auxiliary trace data set, but there is no other tracing overhead.

The format of an exception trace entry is almost identical to that of a normal trace entry. However, you can identify it by the eye-catcher *EXC* in the header.

Note: Exception conditions that are detected by 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 master user trace flag is off. User exception trace entries are always written to the internal trace table (even if internal tracing is set off), but are written to other destinations only if they are active.

The user exception trace entries CICS writes are identified by the character string *EXCU in any formatted trace output produced by CICS utility programs. For
example, an application program exception trace entry generated by an EXEC
CICS ENTER TRACENUM() EXCEPTION command appears in formatted trace
output as:

USER *EXCU - APPLICATION-PROGRAM-EXCEPTION

If you use the exit programming interface (XPI) trace control function to write user
trace entries, you can use the DATA1 block descriptor to indicate whether the entry
is an exception trace entry. Enter the literal `USEREXC' in the DATA1 field on the
DFHTRPTX TRACE_PUT call to identify an exception trace entry. This is
interpreted by the trace formatting utility program as follows:

USER *EXCU - USER-EXIT-PROGRAM-EXCEPTION

See the CICS Customization Guide for programming information about XPI trace
control function.

---

**CICS XRF tracing**

CICS XRF tracing is always active when you are running with XRF. It is used by the
CICS availability manager (CAVM), and you cannot turn it off. However, it only
makes about 12 entries every 2 seconds, so the overhead is not great. Note that
CICS XRF tracing is quite distinct from the "normal" CICS tracing that can originate
from the CAVM, which is identified by trace point IDs AP 00C4 through AP 00C7.

The XRF trace entries are 32 bytes long and are written to a trace table in main
storage. The table has a fixed size of 64KB, and it wraps around when it is full.

The table starts with 28 bytes of control information, in the format shown in

<table>
<thead>
<tr>
<th>Table 22. Control information at the start of the XRF trace table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
</tr>
<tr>
<td>0–15</td>
</tr>
<tr>
<td>16–19</td>
</tr>
<tr>
<td>20–23</td>
</tr>
<tr>
<td>24–27</td>
</tr>
</tbody>
</table>

Trace entries are 32 bytes long, and have the format shown in

<table>
<thead>
<tr>
<th>Table 23. Format of an XRF trace entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2–3</td>
</tr>
<tr>
<td>4–27</td>
</tr>
<tr>
<td>28–31</td>
</tr>
</tbody>
</table>

Process IDs are assigned in order of process ATTACH starting from 1. Some
special values are used for processes which are not known to the dispatcher, but
which cause trace entries to be made. These are:
**Entry types**

The entries are as follows:

*Table 24. XRF trace entry types*

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Subtype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHWLGET</td>
<td>1</td>
<td>1</td>
<td>Module entry Bytes 4-11 Module name Bytes 12-15 LIFO allocation address</td>
</tr>
<tr>
<td>DFHWLFR</td>
<td>1</td>
<td>2</td>
<td>Module return Bytes 4-11 Module name Bytes 12-15 LIFO allocation address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bytes 16-27 0</td>
</tr>
<tr>
<td>DFHWDATT</td>
<td>2</td>
<td>1</td>
<td>XRF process attach Bytes 4-7 Process entry point Bytes 8-11 Initial data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>parameter Bytes 12-15 Address of ESPIE routine Bytes 16-19 Address of ESTAE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>routine Bytes 20-23 Address of attached process XPB Bytes 24-27 Process ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>attached process XPB</td>
</tr>
<tr>
<td>DFHWDISP</td>
<td>2</td>
<td>2</td>
<td>XRF process detach Bytes 4-27 0</td>
</tr>
<tr>
<td>DFHWDISP</td>
<td>2</td>
<td>3</td>
<td>XRF process dispatch Bytes 4-7 Address of external ECB waited for Bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8-11 Address of internal ECB waited for Bytes 12-15 Awaited broadcast</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>events which were posted Bytes 16-19 Broadcast events still posted for this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>process Bytes 20-23 Address of process XPB Bytes 24-27 Process ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>attached process XPB</td>
</tr>
<tr>
<td>DFHWDWAT</td>
<td>2</td>
<td>4</td>
<td>XRF process wait (event data) Bytes 4-7 Address of external ECB to wait for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bytes 8-11 Address of internal ECB to wait for Bytes 12-15 Broadcast</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>events to wait for Bytes 16-19 Events to be broadcast to all processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bytes 20-23 Events to be reset for this process Bytes 24-27 0</td>
</tr>
<tr>
<td>DFHWDAT</td>
<td>2</td>
<td>5</td>
<td>XRF process wait (lock data) Bytes 4-7 Locks to be freed Bytes 8-11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Locks to be acquired Bytes 12-19 0 Bytes 20-23 Locks held by all other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>processes at time of call Bytes 24-27 Locks held by this process at time of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>call</td>
</tr>
<tr>
<td>DFHWDISP</td>
<td>2</td>
<td>6</td>
<td>Dispatcher termination Bytes 4-27 0</td>
</tr>
<tr>
<td>DFHDISP</td>
<td>2</td>
<td>7</td>
<td>Dispatcher issuing OS WAIT Bytes 4-19 0 Bytes 20-23 Address of WAIT list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bytes 24-27 Number of ECBs in WAIT list</td>
</tr>
<tr>
<td>DFHWDISP</td>
<td>2</td>
<td>8</td>
<td>Dispatcher resume after OS WAIT Bytes 4-27 0</td>
</tr>
<tr>
<td>DFHWMMT</td>
<td>3</td>
<td>1</td>
<td>Message manager issuing VSAM GET Bytes 4-7 RPL address Bytes 8-11 RBA of CI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to be read Bytes 12-27 0</td>
</tr>
<tr>
<td>DFHWMMT</td>
<td>3</td>
<td>2</td>
<td>Message manager issuing VSAM PUT Bytes 4-7 RPL address Bytes 8-11 RBA of CI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to be read Bytes 12-27 0</td>
</tr>
</tbody>
</table>

**Process ID Function**

X'0000' Initial attach
X'FFFFE' ESPIE/ESTAE error handling
X'FFFFF' Dispatcher activities.
Table 24. XRF trace entry types (continued)

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Subtype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHWMMT</td>
<td>3</td>
<td>3</td>
<td>Message manager I/O complete Bytes 4- 7 RPL address Bytes 8-11 RBA of CI to be read Byte 12 0 Bytes 13-15 VSAM feedback information Bytes 16-27 0</td>
</tr>
<tr>
<td>DFHWMQH</td>
<td>4</td>
<td>1</td>
<td>Message manager message received Bytes 4- 7 0 Bytes 8-11 Queue name Bytes 12-15 Message sequence number Bytes 16-19 Address of message block (contains message copy) Bytes 20-27 0</td>
</tr>
<tr>
<td>DFHWMWR</td>
<td>4</td>
<td>2</td>
<td>Message manager message sent Bytes 4- 7 0 Bytes 8-11 Queue name Bytes 12-15 Message sequence number Bytes 16-19 Message file cycle number(contains message copy) Bytes 20-23 RBA of this message Bytes 24-25 0 Bytes 26-27 Response to PUTMSG request (WMSRESP)</td>
</tr>
</tbody>
</table>

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 2 on page 42

CICS VTAM exit tracing

CICS VTAM exit tracing gives you a way of tracing VTAM requests made from CICS. You can control it online, using CETR—see Figure 21 on page 234 for an illustration of the screen you need to use.

When CICS issues a VTAM request, VTAM services the request asynchronously and CICS continues executing. When VTAM has finished with the request, it returns control to CICS by driving a CICS VTAM exit. Every such exit contains a trace point, and if CICS VTAM 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 VTAM 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 VTAM 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 VTAM system programming to be able to use it. CICS VTAM exit tracing also has the advantage of tracing some important CICS data areas relating to VTAM requests, which might be useful for diagnosing problems.

**Controlling CICS VTAM exit tracing**

You can turn CICS VTAM 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 VTAM network. However, you cannot select which CICS exits are to be traced. Whenever CICS VTAM exit tracing is running, you get a trace entry every time a CICS exit is driven by VTAM.

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

If you need to turn on CICS VTAM 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 VTAM trace entries are always written to the GTF trace data set, and you can format them in the usual way—see [Chapter 16, “Formatting and interpreting trace entries,” on page 251](#). Direct all “normal” CICS tracing to the GTF trace destination as well, so you get the regular trace entries and the CICS VTAM 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 VTAM exit trace entries**

CICS VTAM 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 [Figure 25 on page 254](#). 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 VTAM 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 [CICS Trace Entries](#).

**VTAM buffer tracing**

VTAM buffer tracing enables you to look at all the data that is passed between logical units on a VTAM communication link.

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 VTAM. For details of VTAM buffer tracing, see the appropriate manual in the VTAM library.
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.

For each component, you can specify two sets of trace level attributes. The trace level attributes define the trace point IDs to be traced for that component when standard task tracing is being done and when special task tracing is being done, respectively.

If you are running a test region, you probably have background tracing most of the time. In this case, the default tracing options (standard tracing for all transactions, and level-1 trace points only in the standard set for all components) probably suffice. All you need do is to enable the required trace destinations and set up any related tracing options. Details are given in "Selecting trace destinations and related options" on page 244.

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 228.

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:

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:
   - for each suspected component, decide whether you need special level-1 tracing only, or level-1 and level-2 tracing
   - turn special tracing off for all other components
3. If you do not need standard tracing, turn the master 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 25.

### Table 25. The combination of task trace options

<table>
<thead>
<tr>
<th>OPTION on TRANSACTION</th>
<th>OPTION on TERMINAL</th>
<th>Task tracing</th>
</tr>
</thead>
<tbody>
<tr>
<td>tracing suppressed</td>
<td>standard tracing</td>
<td>SUPPRESSED</td>
</tr>
<tr>
<td>tracing suppressed</td>
<td>special tracing</td>
<td>SUPPRESSED</td>
</tr>
<tr>
<td>standard tracing</td>
<td>standard tracing</td>
<td>STANDARD</td>
</tr>
<tr>
<td>standard tracing</td>
<td>special tracing</td>
<td>SPECIAL</td>
</tr>
<tr>
<td>special tracing</td>
<td>standard tracing</td>
<td>SPECIAL</td>
</tr>
<tr>
<td>special tracing</td>
<td>special tracing</td>
<td>SPECIAL</td>
</tr>
</tbody>
</table>

You can set up the task tracing you want using the CETR transaction, with the screen shown in Figure 21. 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 VTAM exit tracing for the terminal. For more details about CICS VTAM exit tracing, see "CICS VTAM exit tracing" on page 231.

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.

**Notes:**
1. You can turn standard tracing off for all tasks by setting the master system trace flag off. You can do this with the CETR transaction, using the screen shown in Figure 23 on page 247, or you can code SYSTR=OFF at system initialization. However, any special task tracing will continue—it is not affected by the setting of the system master trace flag.
2. If you run with standard tracing turned off and you specify levels of tracing for the required components under the "Special" heading in the "Components Trace Options" screen shown in Figure 22 on page 238, 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 21 on page 234.

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 26. It is assumed that at least one trace destination is STARTED.

Table 26. Logic used to determine if a trace call is to be made from a trace point

<table>
<thead>
<tr>
<th>Is tracing suppressed for this task?</th>
<th>Is standard tracing required for this task?</th>
<th>Is the master system trace flag on?</th>
<th>Is special tracing specified for this domain and trace level?</th>
<th>Is standard tracing specified for this domain and trace level?</th>
<th>Is trace call made?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>not applicable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>not applicable</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>not applicable</td>
<td>not applicable</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>not applicable</td>
<td>Yes</td>
<td>not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>not applicable</td>
<td>No</td>
<td>not applicable</td>
<td>No</td>
</tr>
</tbody>
</table>

Selecting tracing by component

“Component names and abbreviations” on page 236 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 22 on page 238).

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.

The component codes BF, BM, BR, CP, DC, DI, EI, FC, IC, IS, KC, PC, SC, SZ, TC, TD, TS, UE, and WB are subcomponents of the AP domain. The corresponding trace entries are produced with a point ID of AP nnnn.

For example, trace point AP 0471 is a file control level-1 trace point and AP 0472 is a file control level-2 trace point. These trace points are produced only if the trace setting for the FC component is ø(1,2)” or øALL”. 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.

The SJ domain, which controls JVM tracing, is a special case. As well as using the normal trace levels, the SJ domain uses trace levels 29–32, which are reserved to indicate the JVM trace levels 0, 1, and 2, plus a user-definable JVM trace level. You can activate trace levels 29–32 using the normal system initialization parameters that you would use to set trace levels for components, but to activate these trace levels using the CETR transaction, you need to use the JVM Trace screens, rather than the Component Trace screens. The JVM trace options are set using a “free-form” 240–character field, and you can set these using the JVM tracing system initialization parameters, or the JVM Trace CETR screens. “Defining tracing for JVMs” on page 238 describes how to define JVM tracing.
## Component names and abbreviations

<table>
<thead>
<tr>
<th>Code</th>
<th>Component name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Application domain</td>
</tr>
<tr>
<td>BA</td>
<td>Business application manager</td>
</tr>
<tr>
<td>BF</td>
<td>Built-in function</td>
</tr>
<tr>
<td>BM</td>
<td>Basic mapping support</td>
</tr>
<tr>
<td>BR</td>
<td>3270 bridge</td>
</tr>
<tr>
<td>CP</td>
<td>Common programming interface</td>
</tr>
<tr>
<td>DC</td>
<td>Dump compatibility layer</td>
</tr>
<tr>
<td>DD</td>
<td>Directory manager domain</td>
</tr>
<tr>
<td>DH</td>
<td>Document Handler</td>
</tr>
<tr>
<td>DI</td>
<td>Batch data interchange</td>
</tr>
<tr>
<td>DM</td>
<td>Domain manager domain</td>
</tr>
<tr>
<td>DP</td>
<td>Debugging profiles domain</td>
</tr>
<tr>
<td>DS</td>
<td>Dispatcher domain</td>
</tr>
<tr>
<td>DU</td>
<td>Dump domain</td>
</tr>
<tr>
<td>EI</td>
<td>Exec interface</td>
</tr>
<tr>
<td>EJ</td>
<td>Enterprise Java domain</td>
</tr>
<tr>
<td>EM</td>
<td>Event manager domain</td>
</tr>
<tr>
<td>FC</td>
<td>File control</td>
</tr>
<tr>
<td>GC</td>
<td>Global catalog domain</td>
</tr>
<tr>
<td>IC</td>
<td>Interval control</td>
</tr>
<tr>
<td>IE</td>
<td>ECI over TCP/IP domain</td>
</tr>
<tr>
<td>II</td>
<td>IIOP domain</td>
</tr>
<tr>
<td>IS</td>
<td>ISC or IRC</td>
</tr>
<tr>
<td>KC</td>
<td>Task control</td>
</tr>
<tr>
<td>KE</td>
<td>Kernel</td>
</tr>
<tr>
<td>LC</td>
<td>Local catalog domain</td>
</tr>
<tr>
<td>LD</td>
<td>Loader domain</td>
</tr>
<tr>
<td>LG</td>
<td>Log manager domain</td>
</tr>
<tr>
<td>LM</td>
<td>Lock domain</td>
</tr>
<tr>
<td>ME</td>
<td>Message domain</td>
</tr>
<tr>
<td>MN</td>
<td>Monitoring domain</td>
</tr>
<tr>
<td>NQ</td>
<td>Enqueue domain</td>
</tr>
<tr>
<td>OT</td>
<td>Object transaction domain</td>
</tr>
<tr>
<td>PA</td>
<td>Parameter domain</td>
</tr>
<tr>
<td>PC</td>
<td>Program control</td>
</tr>
<tr>
<td>PG</td>
<td>Program manager domain</td>
</tr>
<tr>
<td>PI</td>
<td>Pipeline domain</td>
</tr>
<tr>
<td>PT</td>
<td>Partner domain</td>
</tr>
<tr>
<td>RI</td>
<td>RMI</td>
</tr>
</tbody>
</table>

236  CICS TS for z/OS: CICS Problem Determination Guide
Defining component tracing at system initialization

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

- **SPCTR** to indicate the level of special tracing required for CICS as a whole.
- **SPCTRxx**, where xx is one of the two-character component identifiers that specify the level of special tracing you require for a particular CICS component (see "Component names and abbreviations" on page 236).
- **STNTR** to indicate the level of standard tracing required for CICS as a whole.
- **STNTRxx**, where xx is one of the two-character component identifiers that specify the level of standard tracing you require for a particular CICS component (see "Component names and abbreviations" on page 236).

For more information about system initialization parameters, see the CICS System Definition Guide.

Defining component tracing when the CICS system is running

You can use the CETR transaction to define component tracing dynamically on the running CICS system.

Figure 22 on page 238 shows you what the CETR Component Trace Options screen looks like. To make changes, you overtype the settings shown on the screen, and then press ENTER.
With the settings shown, trace entries are made as follows:

- With standard task tracing in effect, from level-1 trace points of all the components listed.
- With standard task tracing in effect, from level-2 trace points for the 3270 Bridge component.
- With special task tracing in effect:
  - From level-1 trace points only for components DI, EI, IC, and KC
  - From both level-1 and level-2 trace points for components AP, CP, DD, DM, DS, DU, FC, GC, and KE.

No special task tracing is done for components BF, BM, DC, and IS.

This CETF screen should not be used to define trace levels 29–32 for the SJ domain, which are used to control JVM tracing. "Defining tracing for JVMs" tells you about the screens that you should use for these trace levels.

### Defining tracing for JVMs

The SJ domain uses trace levels 29–32 to control Java Virtual Machine (JVM) trace. These correspond to the CICS options for JVM Level 0 trace, JVM Level 1 trace, JVM Level 2 trace, and JVM User trace. The CICS default settings for these trace options map to trace point levels 0, 1, and 2 for JVMs, and also provide a user-definable trace level which can be used to specify deeper levels of tracing or complex trace options. The JVM trace options are defined using a "free-form" 240-character field. Note that JVM trace can produce a large amount of output, so you should normally activate JVM trace for special transactions, rather than turning it on globally for all transactions.

"Controlling tracing for JVMs" in *Java Applications in CICS* tells you about the different ways to activate JVM tracing and change the JVM trace options. To summarize, you can control JVM trace using:
• The CICS-supplied transaction CETR, which you can use to change the JVM trace options, and to activate JVM tracing. You can use the CETR transaction to define JVM tracing dynamically on the running CICS system.

• The CICS system initialization parameters JVMLEVEL0TRACE, JVMLEVEL1TRACE, JVMLEVEL2TRACE, and JVMUSERTRACE, which you can use to set the default JVM trace options for the CICS region, and the SPCTRSJ and STNTRSJ system initialization parameters, which you can use to activate JVM tracing. You can only supply these parameters at CICS startup time; you cannot define them in the DFHSIT macro. You can use this method to define tracing for JVMs at CICS initialization.

• The EXEC CICS INQUIRE JVMPOOL and EXEC CICS SET JVMPOOL commands (but not their CEMT equivalents), which you can use to change the current JVM trace options for the CICS region.

• The EXEC CICS SET TRACETYPE command, which you can use to activate JVM tracing.

• The \texttt{ibm.dg.trc.external} system property in the JVM properties file that is referenced by the JVM profile for a JVM, which you can use to set and activate initial trace options for a particular JVM. You should only use this system property with care.

The first two methods, using CETR and using the CICS system initialization parameters, are most similar to the methods that you would use to define tracing for other components.

When you set trace levels 29–32 for the SJ component and activate JVM trace, each JVM trace point that is generated appears as an instance of CICS trace point SJ 4D01. If the JVM trace facility fails, CICS issues the trace point SJ 4D00.

The \textit{IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide}, SC34-6358, which is available to download from www.ibm.com/developerworks/java/jdk/diagnosis/, has more detailed information about JVM trace and about problem determination for JVMs. This topic summarizes the information from that document which is most significant for tracing JVMs in CICS. The topic gives information about the JVM trace options that you can set using the JVM Level 0 trace, JVM Level 1 trace, JVM Level 2 trace, and JVM User trace options, and the \texttt{ibm.dg.trc.external} system property.

In addition to the JVM trace options, the standard trace points for the SJ (JVM) domain, at CICS trace levels 0, 1 and 2, can be used to trace the actions that CICS takes in setting up and managing JVMs and the shared class cache. These trace points can be activated using the CETR Component Trace screens, as described in “Selecting tracing by component” on page 235. The SJ domain includes a level 2 trace point SJ 0224, which shows you a history of the programs that have used each JVM. \textit{SJVM domain trace points” in CICS Trace Entries} has details of all the standard trace points in the SJ domain.

### Selecting JVM trace options

When you set trace options for a JVM using the \texttt{ibm.dg.trc.external} system property in the JVM properties file, those options are present, and activated, throughout the JVM’s lifetime, unless they are overridden by a subsequent instruction. That system property should therefore be used with caution, because the amounts of output produced in this way can be very large.

When you activate trace options for a transaction in a JVM using CETR, the SPCTRSJ and STNTRSJ system initialization parameters, or the EXEC CICS SET
TRACETYPE command, CICS uses the JVM's JVMRI (formerly known as JVMRAS) interface to pass the trace options to the JVM at the point when the transaction begins to use the JVM. CICS concatenates the trace options that you have specified using the JVM Level 0 trace, JVM Level 1 trace, JVM Level 2 trace, and JVM User trace options, and prefixes the concatenated option string with EXTERNAL=. The CICS level 2 trace point SJ 052E shows the option string that has been passed to the JVM. This instruction applies the trace options in the JVM as if they had been specified by the ibm.dg.trc.external system property for the JVM, but it applies them only for the duration of the transaction's use of the JVM. The amount of output produced by this method can still be large, but is not so large as the output produced when you set the ibm.dg.trc.external system property directly.

You can select the JVM trace points that you require by using the ibm.dg.trc.external system property, the "free-form" 240–character fields in CETR, or the JVMxxxxTRACE system initialization parameters, to specify some or all of the following parameters:

- A trace point level.
- A component, which can be either a JVM sub-component or a pre-defined group of trace points.
- A trace point type, which can be selected within a component or applied to all components.
- A trace point ID.

The format that you can use to specify these parameters is the same whether you use the ibm.dg.trc.external system property directly, or whether you use one of the methods available in the CICS interface. This topic explains what these parameters are, and gives some examples of how to specify them.

**JVM trace point levels:**

The JVM trace points are classified into levels 0 through 9, based on the importance of the trace point. A level 0 trace point is very important, and this classification is reserved for extraordinary events and errors. A level 9 trace point provides in-depth component detail.

You can specify a given level of tracing by using the level0 through level9 keywords (which can be abbreviated to l0 through l9, or in CICS, can be specified in upper case). Note that enabling a JVM trace point level enables that level and all levels above it. For example, if level 5 is selected, all trace points that have levels 0 through 5 are included. If no trace point level is specified, the default is level 9, meaning that all levels of trace points are provided.

The default JVM trace options that are provided in CICS use the JVM trace point level specifications. The options for JVM Level 0 trace, JVM Level 1 trace, and JVM Level 2 trace specify LEVEL0, LEVEL1, and LEVEL2 respectively, so they map to the Level 0, Level 1 and Level 2 trace point levels for JVMs. It is suggested that you keep these level specifications, but if you find that another JVM trace point level is more useful for your purposes than one of the default levels, you could change the level specification to map to your preferred JVM trace point level (for example, you could specify LEVEL5 instead of LEVEL2 for the JVMLEVEL2TRACE option). Note that enabling a JVM trace point level enables that level and all levels above it, so for example, if you activate JVM Level 1 trace for a particular transaction, you receive Level 0 trace points for that transaction as well. This
means that if you are using only the level specifications as your JVM trace options, you only need to activate the deepest level of tracing that you require, and the other levels are activated as well.

When the JVM trace point level specifications are used on their own, the default of ALL is assumed for the other parameters, so all trace points are enabled at the levels you have selected. You can add to the basic level specifications if you want to include or exclude particular components or trace point types at the selected trace levels. If you modify the basic level specifications in this way, remember that you might need to activate more than one of the JVMLEVEL0TRACE, JVMLEVEL1TRACE, and JVMLEVEL2TRACE options to obtain all the trace points you require. If you want to create more complex specifications for JVM tracing which use multiple trace point levels, or if you do not want to use trace point levels at all in your specification, use the JVMUSERTRACE option to create your option string.

The rest of this topic explains how to identify components, trace point types and trace point IDs, and then gives examples of how to use those parameters and the trace point levels to create customized trace option specifications.

**Components:**

A component, for the purposes of JVM tracing, is either a JVM sub-component or a pre-defined group of trace points.

JVM sub-components are functional areas, like the CICS domains. They are identified by alphabetic codes; for example, cl is classes, dc is data conversion, lk is locking, and st is storage. The chapter on tracing Java applications and the JVM in the IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide, SC34-6358, has the full list of sub-components and explains the meaning of the most important of them. The list of sub-components is subject to change, so you should check the Diagnostics Guide for the latest information, but a recent list is as follows: ci, cl, dc, dg, lk, st, xe, xm, hpi, java, awt, awt_dnd_datatransfer, Audio, jit, jdwp, mt, fontmanager, net, awt_java2d, awt_print, core, nio.

Alternatively, a component can be a pre-defined group of trace points. They are also identified by alphabetic codes; for example, st_dump identifies the group of trace points in the storage sub-component that relate to dumps, and nativeMethods identifies the group of trace points that relate to native methods. The available groups are listed in the chapter on tracing Java applications and the JVM in the IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide, SC34-6358. The groups are subject to change, so you should check the Diagnostics Guide for the latest information, but a recent list is as follows: checkedjni, compiledMethods, jitCl, jitCompCtrl, jitEh, jitError, jitInit, jniSt, jni, jvmmni, nativeMethods, st_alloc, st_backtrace, st_calloc, st_compact, st_compact_dump, st_compact_verbose, st_concurrent, st_concurrent_pck, st_concurrent_shadow_heap, st_dump, st_freelist, st_icompact, st_loaAdjustTargetSize, st_mark, st_parallel, st_refs, st_terse, st_trace, st_vbornege, st_verify, st_verify_heap, staticMethods, threadSR, wrappedjni.

**JVM trace point types:**

The JVM trace points are classified into the following types, which are identified by name:

- Event
You can specify a trace point type within a particular component, or you can specify a type to be shown for all components (by using the ALL option instead of a component name).

**JVM trace point IDs:**

Each JVM trace point has a numeric ID, which can be specified as part of a trace option string, using the `tpid` keyword, in order to select or exclude individual trace points. Note that trace point level specifications do not apply to trace points that are explicitly specified by their trace point ID. This means that you can add trace point IDs that are classified at any level to any of the JVM trace options in CICS, and they will be supplied when the option is activated, regardless of the trace point levels that are enabled by that trace option.

The TraceFormat.dat file, supplied with the IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2, shows the level (0–9), type, ID and name of each JVM trace point, the JVM sub-component with which it is associated, and the template containing the description of the trace point. Note that this file is subject to change without notice; a version number is included as the first line of the file, and this will be updated if the file is changed. By default, this file is in the library `/usr/lpp/java142/J1.4/lib/`, where `/java142/J1.4/` is the directory that is defined when you install IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2. You can use this file to identify an individual JVM trace point.

**Format and examples:**

The format for specifying the **component, type and trace point ID** in an option string (using either the `ibm.dg.trc.external` system property or one of the methods available in the CICS interface) is as follows:

```
[!]component[[type,...]]
[!]tpid(tracepoint_id,...)
```

The rules for specifying these parameters are:

- `!` is a logical not, which can be applied to a component or trace point ID. The component or trace points that are specified immediately following the `!` are turned off.
- `ALL` is the default in each case if you do not specify a component, type or trace point ID explicitly.
- Trace point IDs must be specified in hexadecimal. You can omit leading zeroes. You can specify a range of trace points by using a hyphen (dash, minus); for example, `tpid(18007,c003-c01f)`.

The following examples show how you might combine these parameters:

- `ALL, IDC, ILK` All trace points except those in the DC and LK components
- `CL(ENTRY,EXIT)` All entry and exit trace points in the CL component
- `ST, !TPID(4001,4002)` All trace points in the ST component except 4001 and 4002
- `ALL(ENTRY,EXIT)` All entry and exit trace points (in all components)

A **JVM trace point level** can be used on its own, which is the case for the default level specifications in CICS. In this case, the default of `ALL` is assumed for the other parameters, so all trace points at and above that level are enabled. Alternatively, a
trace point level can be applied to all or part of your trace point selection, or to an individual component. It does not apply to trace points that are explicitly specified by their trace point ID. You can use more than one trace point level in an option string. For any complex option string, you should use the JVMUSERTRACE option in CICS. The JVMLEVEL0TRACE, JVMLEVEL1TRACE, and JVMLEVEL2TRACE options should each be used for a single trace point level.

You can add components to the default level specifications in CICS if you want to limit JVM tracing to particular components. For example, you could add components to the default JVMLEVEL1TRACE option to specify the following option string:

```
LEVEL1,ST,CL
```

This option string enables trace points that have a level of 0 or 1 for the ST and CL components only. You can also exclude a component by using the logical not (!). For example:

```
LEVEL1,ALL,!LK
```

enables trace points that have a level of 0 or 1 for all components except the LK component. You can include trace point types in your option string as well as components. For example:

```
LEVEL1,ALL(ENTRY,EXIT)
```

enables, in all components, entry and exit trace points that are classified as level 0 or 1.

You can also create complex option strings that use more than one trace point level. When the keyword for a trace point level is used before a trace point selection, that keyword applies to all trace point selection criteria that follow it in the option string. For example, the following option string:

```
LEVEL5,ST,LK,CL,LEVEL1,ALL
```

(which can be abbreviated to)

```
L5,ST,LK,CL,CL,L1,ALL
```

enables trace points that have a level of 0 through 5 for the ST, LK, and CL components and that have a level of 0 or 1 for all other components.

When the keyword for a trace point level is used in brackets after an individual component, it applies only to the component for which it is specified. For example:

```
ST(L5),LK(L5),CL(L5),ALL(L1)
```

enables the same levels of trace points as the previous example.

If you use the logical not operator (!) with a component name when you specify a trace point level in brackets, the result is that the level is inverted, and all trace points deeper than the specified level are excluded. For example, specifying

```
!ST(L5)
```

disables all trace points at levels 6, 7, 8 and 9 for the ST component.

**Interpreting JVM trace**

JVM trace appears as CICS trace points in the JVM domain. When you activate JVM trace, each JVM trace point that is generated appears as an instance of CICS trace point SJ 4D01. The following example shows how a JVM trace point appears in CICS trace that has been formatted using the FULL option:
The information in the trace data is interpreted as follows:

**Bytes 0–3 (in the example, 49B8B2A8, shown as 'ThreadID')**
A unique address representing the thread.

**Byte 4 (in the example, D7, interpreted as part of 'ThreadID')**
Either P or *, P meaning that the thread is prime (CICS TCB), and * meaning that the thread is other than prime.

**Bytes 6–8 (in the example, 00C0 94, shown as 'TPID')**
The JVM trace point ID. This ID identifies a record in the TraceFormat.dat file supplied with the IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2. The TraceFormat.dat file shows the level (0–9), type, ID and name of each JVM trace point, the JVM sub-component with which it is associated, and the template containing the description of the trace point. Note that this file is subject to change without notice; a version number is included as the first line of the file, and this will be updated if the file is changed. By default, this file is in the library /usr/lpp/java142/J1.4/lib/, where /java142/J1.4/ is the directory that is defined when you install IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2.

**Byte 9 and following (in the example, displayed as "main", "com/ibm/cics/server/Wrapper")**
Inserts (if any) for the template associated with the JVM trace entry. Strings are delimited by " and converted from ASCII to EBCDIC. Locate the trace point and its template in the TraceFormat.dat file supplied with the IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2. The template contains the description of the trace point, and the insertion points for the inserts are marked with a percentage sign (%) together with an alphabetic data type code. In the example, the trace point 00C094 has the template ôxeRunJniMethod() About to execute JNI method: %s for class %s", so the full text of the message is ôxeRunJniMethod() About to execute JNI method: main for class com/ibm/cics/server/Wrapper".

### Selecting trace destinations and related options

The trace destinations you select and the options you define for the destinations depends on your specific problem determination requirements. You can select any combination of CICS internal tracing, CICS auxiliary tracing, and CICS GTF tracing. Your decision must be based on the characteristics of the various types of CICS tracing, which are described in "CICS internal trace" on page 245, "CICS auxiliary trace" on page 245, and "CICS GTF trace" on page 246. You need to decide how much trace data you need to capture, and whether you want to integrate CICS tracing with tracing done by other programs.

You can control the status and certain other attributes of the various types of CICS tracing either dynamically, using the CETR transaction, or during system initialization, by coding the appropriate system initialization parameters.
**CICS internal trace**

You can select a status of STARTED or STOPPED for internal trace. If you select STARTED, any trace calls that are made cause trace entries to be directed to the internal trace table. This occupies a sequence of pages in virtual storage above the 16MB line, in MVS GETMAIN storage.

The internal trace table has a minimum size of 16KB, and a maximum size of 1A048A576KB. The table is extendable from 16KB in 4KB increments. You can change the size of the table dynamically, while CICS is running, but if you do so you lose all of the trace data that was present in the table at the time of the change. If you want to keep the data and change the size of the table, take a system dump before you make the change.

The internal trace table wraps when it is full. When the end of the table is reached, the next entry to be directed to the internal trace entry goes to the start, and overlays the trace entry that was formerly there. In practice, the internal trace table cannot be very big, so it is most useful for background tracing or when you do not need to capture an extensive set of trace entries. If you need to trace CICS system activity over a long period, or if you need many entries over a short period, one of the other trace destinations is likely to be more appropriate.

Note that the internal trace table is always present in virtual storage, whether you have turned internal tracing on or not. The reason is that the internal trace table is used as a destination for trace entries when CICS detects an exception condition. Other trace destinations that are currently selected get the exception trace entry as well, but the entry always goes to the internal trace table even if you have turned tracing off completely. This is so that you get "first failure data capture".

**CICS auxiliary trace**

CICS auxiliary trace entries are directed to one or other of two auxiliary trace data sets, DFHAUXT and DFHBUXT. These are CICS-owned BSAM data sets, and they must be created before CICS is started. They cannot be redefined dynamically.

You can use the AUXTR system initialization parameter to turn CICS auxiliary trace on or off in the system initialization table.

You can select a status of STARTED, STOPPED, or PAUSED for CICS auxiliary trace dynamically using the CETR transaction. These statuses reflect both the value of the auxiliary trace flag, and the status of the current auxiliary trace data set, in the way shown in Table 27.

<table>
<thead>
<tr>
<th>Auxiliary tracing status</th>
<th>Auxiliary trace flag</th>
<th>Auxiliary trace data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTED</td>
<td>ON</td>
<td>OPEN</td>
</tr>
<tr>
<td>STOPPED</td>
<td>OFF</td>
<td>CLOSED</td>
</tr>
<tr>
<td>PAUSED</td>
<td>OFF</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

When you first select STARTED for CICS auxiliary trace, any trace entries are directed to the initial auxiliary trace data set. If CICS terminated normally when auxiliary trace was last active, this is the auxiliary trace data set that was not being used at the time. Otherwise, it is the DFHAUXT data set. If you initialize CICS with auxiliary trace STARTED, DFHAUXT is used as the initial auxiliary trace data set.
What happens when the initial data set is full depends on the status of the auxiliary switch (AUXTRSW). This can have a value of NO, NEXT, or ALL, and you can define it either in the system initialization table, using the AUXTRSW system initialization parameter, or dynamically using the CETR transaction.

**NO** means that when the initial data set is full, no more auxiliary tracing is done.

**NEXT** means that when the initial data set is full, then the other data set receives the next trace entries. However, when that one is full, no more trace data is written to auxiliary trace.

**ALL** means that auxiliary trace data is written alternately to each data set, a switch being made from one to the other every time the current one becomes full. This means that trace entries already present in the trace data sets start getting overwritten when both data sets become full for the first time.

The advantage of using auxiliary trace is that you can collect large amounts of trace data, if you initially define large enough trace data sets. For example, you might want to do this to trace system activity over a long period of time, perhaps to solve an unpredictable storage violation problem.

A time stamp is included in the header line of every page of abbreviated auxiliary trace output to help match external events with a particular area of the trace, and thus help you to find the trace entries that are of interest.

**CICS GTF trace**

CICS GTF trace is an invocation of MVS GTF trace, and the trace entries are directed to a destination defined to MVS. This can be either a trace table in main storage, or a single external GTF trace data set.

You can switch CICS GTF trace on or off by using the GTFTR system initialization parameter.

You can select a status of STARTED or STOPPED for CICS GTF trace dynamically using the CETR transaction. MVS GTF trace must be started with the TRACE=USR option before CICS GTF trace is started, because otherwise no trace entries can be written.

In a multi-system environment, trace entries from CICS Version 3, CICS for MVS/ESA 4.1, CICS Transaction Server for z/OS, Version 2 Release 3, and CICS Transaction Server for z/OS, Version 3 Release 1, can be recorded in the GTF trace data set. Entries from all releases are formatted by the CICS-supplied routine, DFHTG640. See the [CICS Operations and Utilities Guide](http://example.com) for details of how to format and print the GTF trace data set.

When the GTF trace data set is full, it wraps. The next trace entries are written at the start of the data set, and the entries that were formerly there are overlaid. Thus, you need to define a data set that is big enough to capture all the trace entries that interest you.

The MVS GTF trace destination can be used not only by CICS, but by other programs as well. This gives you the opportunity of integrating trace entries from CICS with those from other programs. A single GTF trace data set could, for
example, contain trace entries made by both CICS and VTAM. You can relate the
two types of trace entry using a unique task identifier in the trace header, known to
both CICS and VTAM.

Because different products are likely to execute asynchronously, be cautious about
using the sequence of trace entries in the GTF trace data set as evidence when
you investigate problems.

## Setting the tracing status

You can set the system tracing status by coding the appropriate system initialization
parameters, and you can also set it dynamically, using the CETR transaction. The
CETR transaction enables you to make changes in response to contingencies, as
they arise.

### Setting the tracing status at system initialization

These are the parameters that you can use to set up tracing status at system
initialization:

- **AUXTR** to specify whether auxiliary trace is to be activated at system
  initialization.
- **AUXTRSW** to define the auxiliary switch status.
- **GTFTR** to enable CICS to use MVS GTF tracing.
- **INTTR** to specify whether internal tracing is to be activated at system
  initialization.
- **SYSTR** to set the master system trace flag on or off.
- **TRTABSZ** to specify the size of the internal trace table.
- **USERTR** to set the master user trace flag on or off. It must be on if user trace
calls are to be made from your applications.

For more details of system initialization parameters, see the [CICS System Definition
Guide](https://example.com).

### Setting the tracing status using CETR

You can use the CICS/ESA trace control transaction (CETR) to set the tracing
status. Figure 23 shows you the CETR screen you can use to set the tracing status
dynamically.

In this example, internal tracing status is STOPPED, and so regular tracing is not
directed explicitly to the internal trace table. However, note that the internal trace
table is used as a buffer for the other trace destinations, so it always contains the
most recent trace entry if at least one trace destination is STARTED. The internal trace table is also used as a destination for exception trace entries, which are made whenever CICS detects an exception condition. If such a condition were detected when the options shown in this example were in effect, you would be able to find the exception trace entry in the internal trace table as well as in the GTF trace data set.

The internal trace table size is 16KB, which is the minimum size it can be. If internal trace were STARTED, the trace table would wrap when it became full.

The current auxiliary trace data set is B, meaning that trace entries would be written to DFHBUXT if auxiliary tracing were started. However, its status is shown to be PAUSED, so no tracing is done to that destination. The auxiliary switch status is ALL, so a switch would be made to the other auxiliary trace data set whenever one became full.

The GTF trace status is shown to be STARTED, which means that CICS trace entries are written to the GTF trace data set 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. The master system trace flag is OFF. This means that no standard tracing is done at all, even though standard tracing might be specified for some tasks. However, special task tracing is not affected—the master system trace flag only determines whether standard task tracing is to be done.

You can see the role of the master system trace flag in Table 26 on page 235.

The master user trace flag is OFF, so no user trace entries can be made from applications. You must set the master user trace flag on before any user trace requests in your programs can be serviced. If it were off, any trace call requests in your programs would be ignored.
The following logic is used to ensure that trace entries are written to the required destinations:

1. The trace entry is built in the internal trace table.
2. If auxiliary tracing status is STARTED, the trace data is copied to the current auxiliary trace data set.
3. If GTF tracing status is STARTED and GTF tracing is started under MVS with the TRACE=USR option, the trace data is copied to the GTF trace data set.
Chapter 16. 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. For more details of trace utility programs, see the CICS Operations and Utilities Guide.

You can specify abbreviated, short, or extended 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; extended-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 may 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 extended 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 extended trace entries.

The internal trace table can be formatted in one of two ways:
1. From a CICS system dump, using the CICS print dump exit, DFHPD640.
2. From a transaction dump, using the CICS dump utility program, DFHDU640.

Auxiliary trace can be formatted using the CICS trace utility program, DFHTU640. 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.

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).
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.

There are two slightly different extended trace entry formats. One (“old-style”) resembles the format used in earlier releases of CICS, and gives FIELD A and FIELD B values. The other (“new-style”) uses a completely new format, described below.

Both types of formatted trace entries always show:

- The **trace point ID**. This is an identifier that indicates where the trace point is in CICS code. In the case of application (AP) domain, the request type field included in the entry is also needed to uniquely identify the trace point. For all other domains, each trace point has a unique trace point ID.
  
  Its format is always a two-character domain index, showing which domain the trace point is in, then a space, then a four-digit (two-byte) hexadecimal number identifying the trace point within the domain.
  
  The following are examples of trace point IDs:
  
  - AP 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

- An **interpretation string**, showing:
  - The **module** where the trace point is located
  - The **function** being performed
  - Any **parameters** passed on a call, and any response from a called routine.

- A **standard information string**, showing:
  - The **task number**, which is used to identify a task uniquely for as long as it is in the system. It provides a simple way of locating trace entries associated with specific tasks, as follows:
    
    - A five-digit decimal number shows that this is a trace entry for a task with a TCA, the value being taken from field TCAKCTTA of the TCA.
    
    - A three-character non-numeric value in this field shows that the trace entry is for a system task. You could, for example, see "iii" (initialization), or "tcp" (terminal control).
    
    - A two-character domain index in this field shows that the trace entry is for a task without a TCA. The index identifies the domain that attached the task.
  
  - The **kernel task number** (KE_NUM), which is the number used by the kernel domain to identify the task. The same KE_NUM value for the task is shown in the kernel task summary in the formatted system dump.
  
  - The **time** when the trace entry was made. (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, J8, 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.

  

• A number of **data fields** containing information relevant to the function being performed.

  For old-style trace points, these are shown as fixed length (4-byte) FIELD A and FIELD B values in the same line as the interpretation string. Both the hexadecimal data values and any printable EBCDIC characters that they represent are shown.

  For new-style trace points, 1–7 variable-length data fields can be given. They are shown immediately below the standard information line. Any printable EBCDIC characters represented by byte values in the data fields are shown on the right of the 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. Trace entries from old-style trace points also include a **request type field** (REQ), which gives the same information as the two-byte request field in the formatted trace entries of CICS/MVS Version 2.

  Some old-style trace entries also have a RESOURCE field. When provided, it is usually the name of a resource associated with the request being traced. For example, for program control requests, it is the program name.

  **Figure 24** shows a trace entry made from an old-style trace point. Its trace point ID is AP 00E1, corresponding to trace ID X'E1' of CICS/MVS Version 2. If you are migrating from CICS/MVS Version 2, you can probably see the resemblance between this type of format and the one it replaces.

  

Figure 24. Example of the extended format for an old-style trace entry

The following is an explanation of the old-style trace entry:

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

  Note that although all old-style trace points are in AP domain, not all AP domain trace points are old-style. Some are new trace points, and they have a similar format to that shown in **Figure 25 on page 254**. In general, old-style trace points have values less than or equal to X'00FF' and new AP-domain trace points have values greater than or equal to X'0200'.

• EIP EXIT SYMCOIN 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.

  – SYMCOIN shows the type of function requested.
REQ(00F4) represents the request type of the trace format of CICS/MVS Version 2. In this example, byte 1 bits 0–3 (X'F') show that the trace entry is made on exit from the request.

FIELD-A and FIELD-B contain the same data as FIELD A and FIELD B in the old-style format.

FIELD-A bytes 0–3 would contain the secondary response, EIBRESP2. FIELD-B bytes 0–1 would contain the condition number, EIBRESP. In this example, both are zero, indicating that no error response has been returned. FIELD-B bytes 2–3 contain the command code, EIBFN. In this example, this is X'1602', showing that the EXEC CICS command was SYNCPOINT.

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

The standard information string shows:
- TASK-00163 shows that the task number for the currently running task is 00163. Any trace entries having the same task number would have been made while this task was running.
- KE_NUM-0007 shows that the kernel task number for the task is 0007. If you were to take a system dump while this task was in the system, you could identify the task in the kernel summary information from this number.
- TCB-QR /009F3338 is the address of the MVS TCB
- RET-8413F470 is the return address
- The time when the trace entry was made was 16:32:01.1291125625.
- The interval that elapsed between this and the preceding trace entry was 00.0000216875 seconds.

Look now at Figure 25, which shows a new-style trace entry.

![Figure 25](image)

Figure 25. Example of the extended format for a new-style trace entry

The following is an explanation of the new-style trace entry:
- SM 0C01 shows that this trace entry was made from trace point X'0C01' in the storage manager domain.
- 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 the trace call was made from within module DFHSMMG.
  - ENTRY FUNCTION(GETMAIN) tells you the call was made on entry to the GETMAIN function.
  - GET_LENGTH(1A4A) SUSPEND(YES) INITIAL_IMAGE(00) STORAGE_CLASS(TASK) tells you the parameters associated with the GETMAIN call, as follows:
    - The request is for X'1A4A' bytes of storage.
    - The task is to be suspended if the storage is not immediately available.
    - The storage is to be initialized to X'00'
    - The storage class is TASK.
- The standard information string gives you this information:
The task currently running is task number X'00163'.
The kernel task number for the task is 0007.
The time when the trace entry was made was 16:31:52.5916976250.
The time that elapsed between this and the preceding trace entry was 00.0000666250 seconds.

- The data that is displayed below the standard information was taken from only one data area.
  - 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.

Note that information about some data areas is intended for use by IBM support personnel, and this means that details of their format and contents might not be available to you. If you reach a point at which you are certain that you cannot continue the problem determination process because you do not have access to information about a data area, you need to contact your IBM Support Center. They may not necessarily tell you the format and contents, but they investigate the problem in the usual way, as described in Chapter 20, “IBM program support,” on page 311.

### Interpreting short-format CICS system trace entries

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

Figure 26 shows an example of the short-format for an old-style entry.

<table>
<thead>
<tr>
<th>00030</th>
<th>QR</th>
<th>AP 00E1</th>
<th>EIP ENTRY INQUIRE-TRACEFLAG</th>
<th>REQ(0004)</th>
<th>FIELD-A(071F6018 ..-.)</th>
<th>FIELD-B(08007812 ....)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RET-870844CE</td>
<td>11:39:44.8516351250 80.0000343750 +000011=</td>
</tr>
</tbody>
</table>

Figure 26. 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 FIELD-A and FIELD-B 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
Figure 27 shows an 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-8735CBAC 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.

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

Figure 28 gives an example of the abbreviated format for an old-style trace entry.

In this example:
- 00021 is the task number
- QR is the TCB ID

Interpreting abbreviated-format CICS system trace entries

Abbreviated-format CICS trace entries contain much of the information present in the corresponding extended-format trace entries, and they are often sufficient for debugging purposes. There is a one-to-one correspondence between the trace entry numbers for the abbreviated and extended trace entries, so you can easily identify the trace entry pairs.

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

Figure 28 gives an example of the abbreviated format for an old-style trace entry.
AP 00E1 is the trace point ID
EIP ENTRY INQUIRE-TRACEFLAG is the abbreviated interpretation string
0004 is the request field
00223810 .... is FIELD A, with its EBCDIC interpretation
0007812 .... is FIELD B, 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.

Abbreviated old-style trace entries are easy to interpret, as you can readily identify the REQ, FIELD A and FIELD B fields. Note that some such trace entries also include a RESOURCE field.

For ZCP trace entries, FIELD B (which contains the TCTTE address) is printed twice on each line. This allows both sides of the output to be scanned for the terminal entries on an 80-column screen without having to scroll left and right.

*Figure 29* gives an example of the abbreviated format for a new-style trace entry.

![Figure 29](image-url)

*Figure 29. 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 03BBA370 , 00000001,848659C0,048659A0,410,200,REUSABLE is the abbreviated interpretation string
- =000005= is the trace entry number

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 30* shows the corresponding extended-format trace entry.

![Figure 30](image-url)

*Figure 30. 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 DFHLDDL.
- 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 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.

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 31.

```
AP 000B USER EVENT - APPLICATION-PROGRAM-ENTRY - SEND - CICS USER TRACE ENTRY HELP INFORMATION
    TASK-00163 KE_NUM-0007 TCB-QR /009F3338 RET-B003F54C TIME-16:32:01.1295568750 INTERVAL-00.0001965625 =000731=
    1-0000 E4E2C5D9 40404040 *USER *
    2-0000 C3C9C3E2 40E4E2C5 D940E3D9 C1C3C540 C5D5E3D9 E8404040 40404040 40404040 40404040 *CICS USER TRACE ENTRY *
    3-0000 E2C5D5C4 40404040 *HELP INFORMATION *
```

Figure 31. 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 31 on page 258 is shown in Figure 32.

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 33 shows an example of the short format for a user trace entry.

Figure 32. Example of the abbreviated format for a user trace entry

Figure 33. Example of the short format for a user trace entry
Chapter 17. 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:

- Setting up the dumping environment, so that the appropriate dump action is taken when circumstances arise that might cause a dump to be taken.
- 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 dumps](#).

Setting up the dumping environment

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

- System dumps (apart from CICS kernel domain dumps) can be globally suppressed or enabled at system initialization by using the DUMP system initialization parameter.
- System dumps (apart from CICS kernel domain dumps) can be globally suppressed or enabled dynamically through the EXEC CICS SET SYSTEM DUMPING command.
- Transaction dumps can be suppressed or enabled dynamically for individual transactions. This is done by using the EXEC CICS SET TRANSACTION DUMPING system programming command, or the CEMT SET TRDUMPCODE command or by using the DUMP attribute of the RDO definition for the transaction.
- System dumps (apart from CICS kernel domain dumps) can be suppressed for specific dump codes from a dump domain XDUREQ global user exit program.
- System dumps (apart from CICS kernel domain dumps) can be suppressed or enabled, and other dumping requirements specified, by means of 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 275.
- Dumps of the builder parameter set at specific stages in the build process of terminal or connection definition can be enabled. This is done using the CSFE ZCQTRACE facility. For details, see [The CSFE ZCQTRACE facility](#) on page 283.
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 [OS/390 MVS Diagnosis: Tools and Service Aids](http://www.ibm.com/systems/mvs/390/diag/index.html).

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 transactions 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 master terminal, if system dumping has not been globally suppressed. The system dump code that is needed is supplied by CICS, and it has a specific value of “MT0001”.

- You can use **EXEC CICS PERFORM DUMP** to get a CICS system dump, if system dumping is not globally suppressed. You must specify a system dump code when you use this command and it must be a maximum of 8 characters in length.

- You can use **EXEC CICS DUMP TRANSACTION** to get a transaction dump. You get a transaction dump even if dumping has been suppressed for the transaction that you identify on the command.
You must specify a transaction dump code when you use this command and it must be a maximum of 4 characters in length. It could, for example, be TD01.

- You can make a TRANSACTION_DUMP or a SYSTEM_DUMP XPI call from an exit program to get a transaction dump or a system dump, respectively. For programming information about these exits, see the CICS Customization Guide.

You might use these methods of taking dumps if, for example, you had a task in a wait state, or you suspected that a task was looping. However, these methods are not useful for getting information following a transaction abend or a CICS system abend. This is because the evidence you need is almost certain to disappear before your request for the dump has been processed.

**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 actually taken. CICS does not take a transaction dump if a HANDLE ABEND is active at the current logical level. 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 and Codes. 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 and Codes.

- 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

---

2. To make PL/I on units work, PL/I library routines can issue HANDLE ABEND. This is called an implicit HANDLE ABEND and causes the suppression of transaction dumps.
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 the entry for message DFHZC3496 in CICS Messages and Codes, also read the programming information on NEPs in the CICS Customization Guide.
- A CICS system dump can also be requested from the global trap/trace exit. In this case, the system dump code is TR1003.

CICS dumping in a sysplex

Simultaneous dump capture from multiple CICS regions across a sysplex is possible, considerably aiding problem determination in XCF/MRO environments where many CICS regions are running. Two types of situation, in particular, benefit from this:

- Where a task involves multiple CICS regions in a sysplex, and one region issues a dump, typically in response to an error.
  Dump data from all CICS regions related to the region issuing the dump request is usually required in order to fully diagnose and solve the problem. This dump data from related regions would also need to be captured at the same time as the dump taken on the region issuing the dump.
- Where an MVS console operator needs to capture, simultaneously, dump data from multiple CICS regions in the sysplex.

Before CICS for MVS/ESA 4.1, such automatic and simultaneous dump data capture of CICS data was impossible.

You need MVS/ESA 5.1, the MVS workload manager, and the XCF facility in order to collect dump data in this way. The MVS images in the sysplex must be connected via XCF. The CICS regions must be using MRO supported by the CICS Transaction Server for z/OS, Version 3 Release 1 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 VTAM 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 master terminal and system programming commands:
- EXEC CICS INQUIRE SYSDUMPCODE
- EXEC CICS SET SYSDUMPCODE
- EXEC CICS INQUIRE TRANDUMPCODE
- EXEC CICS SET TRANDUMPCODE
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 may want to issue a dump request simultaneously to several CICS regions in the sysplex. There may 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=()
RX,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** - an MVS keyword. This 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** - an MVS keyword. This 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** - a CICS keyword. The operator should include the generic job name. This is used by DFHDUMPX to determine which address spaces to dump.
See the *MVS System Commands* manual, GC28-1626, for a full description of all command options.

If you adopt a suitable naming convention for your CICS regions, this can be used to define suitable generic jobnames to determine which CICS regions to dump. See the *System/390 MVS Sysplex Application Migration* manual for recommendations on naming conventions. If you follow the recommendation in this manual, 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:

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

**Useful CICS master 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 Transaction Server for z/OS, Version 3 Release 1 level. Connections using VTAM 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 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,COUPL - 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  DEVEN: 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  DEVEN: 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 CICS master terminal command CEMT I CONNECTION to display the status of the connections. 'XCF' is displayed for every acquired connection using MRO/XCF for communications.
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 Transaction Server for z/OS, Version 3 Release 1 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.
In the next example, the messages from SDUMP indicate that one dump of the master address space has been taken.

```
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,
sysdloc1,(dfhjobn,lyahzct))
11.36.49 DEV5  IEE6001 REPLY TO 03 IS;REMOTE=(SYSLIST=+),
               PROBDESC=(SYSDCOND,SYSDL
11.36.52 DEV5  IEA794I SVC DUMP HAS CAPTURED:
               DUMPID=001 REQUESTED BY JOB (*MASTER*)
               DUMP TITLE=NO CICS
```

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

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

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

```
+11.43.42 DEV5  *IEA911E COMPLETE DUMP ON SYS1.DUMP03
+DUMPID=002 REQUESTED BY JOB (*MASTER*)
+FOR ASID (0001)
+REMOTE DUMPS REQUESTED
```

To verify that the remote dumping function works on the local system, use the following commands:
The messages from SDUMP indicate two dumps were taken, one for the master 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.

The following example lists the MVS console messages received when the CICS master 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, 001B, 001E, 001F, 0020, 001D, 0022, 0024, 0021, 0023, 0028, 0025, 0029, 0026 on MVS image DEV7.

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

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).
Chapter 17. Using dumps in problem determination

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. You do this by adding the dump code (constructed by removing the “DFH” prefix from the message number) to the system dump table, and specifying the SYSDUMP option. CICS then causes a system dump to be taken when the message is issued.
To determine which messages you can do this for, look in the CICS Messages and Codes manual. If the message you are interested in has a 2-character alphabetic component ID after the “DFH” prefix, and it has either XMEOUT global user exit parameters or a destination of “Terminal User”, you can use it to construct a system dump code to add to the dump table.

You cannot enable dumping for messages that do not have these characteristics. For example, some messages that are issued early during initialization cannot be used to cause CICS to take a system dump, because the mechanisms that control dumping might not be initialized at that time. Also, you cannot enable dumping for the message domain’s own messages (they are prefixed by “DFHME”) where they do not normally cause CICS to take a system dump.

System dump actions with messages DFHAP0001 and DFHSR0001

In the event of a program check or 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 STG PROT=YES is specified and the hardware and software environment supports the storage protection facility. CICS determines which of these messages to issue depending on whether or not the program check or 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 via 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 non-application 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.

Notes:
1. Only a transaction dump code can cause both a transaction dump and a system dump to be taken.
2. If a severe error is detected, the system can terminate CICS even if you specify that CICS is not to be terminated.
3. Values of 0–998 for “maximum times dump code action can be taken” are literal, but a value of 999 (the default) means there is no limit.
4. You cannot suppress CICS kernel domain dumps.

All the options you specify are recorded in the appropriate dump table, and written in the CICS global catalog. Any dump table entries you have changed or created during a CICS run are preserved when CICS is subsequently shut down. They are available during the next run unless CICS is cold started, when they are deleted from the global catalog.

The only circumstances in which dump table additions and changes are lost are:
- When CICS is cold started.
- When the CICS global catalog is redefined, although this is likely to be done only in exceptional circumstances.
• When CICS creates a temporary dump table entry for you, because you have asked for a dump for which there is no dump code in the dump table.

**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 is incremented every time the dump code action is taken, irrespective of what action has been specified in the dump table entry. The current count can be reset to zero by a CEMT transaction, by a system programming command, or by statistical interval expiry.

If system dumping is globally suppressed:

– The dumps-taken count for a system dump code is not incremented by dump requests using that dump code, but the dumps-suppressed count is incremented.

– The dumps-taken count for a transaction dump code specifying a system dump is incremented by dump requests using that dump code. This is so even if the dump code specifies that only a system dump is to be taken.

• For a transaction dump code, the number of transaction dumps that have been taken. The number is incremented every time a transaction dump is taken for this dump code. However, it is not incremented if transaction dumping has been suppressed in this table entry.

• For a transaction dump code, the number of transaction dumps that have been suppressed.

• The number of system dumps that have been taken. The number is incremented every time a system dump is taken for this dump code. It is not incremented if system dumping has been suppressed, either explicitly in the dump table entry or because system dumping has been suppressed globally.

• The number of system dumps that have been suppressed, either explicitly for this dump code or because system dumping has been suppressed globally.

**Notes:**

1. Dump code statistics are all reset when CICS is shut down—unlike dump code attributes, which are not reset.

2. The following dump code statistics are reset at the end of every statistics collecting interval:

   • Number of transaction dumps taken
   • Number of transaction dumps suppressed
   • Number of system dumps taken
   • Number of system dumps suppressed.
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>
<thead>
<tr>
<th>Action</th>
<th>Default</th>
<th>Attribute</th>
<th>Permitted value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take a transaction dump?</td>
<td>YES</td>
<td>TRANDUMPING</td>
<td>TRANDUMP or NOTRAN DUMP</td>
</tr>
<tr>
<td>Take a system dump?</td>
<td>NO</td>
<td>SYSDUMPING</td>
<td>SYSDUMP or NOSYSDUMP</td>
</tr>
<tr>
<td>Take system dumps on related systems?</td>
<td>NO</td>
<td>DUMPSCOPE</td>
<td>LOCAL or RELATED</td>
</tr>
<tr>
<td>Shut down CICS?</td>
<td>NO</td>
<td>SHUTOPTION</td>
<td>SHUTDOWN or NOSHUTDOWN</td>
</tr>
<tr>
<td>Maximum times dump code action can be taken</td>
<td>999</td>
<td>MAXIMUM</td>
<td>0 through 999</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Action</th>
<th>Default</th>
<th>Attribute</th>
<th>Permitted value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take a system dump?</td>
<td>YES</td>
<td>SYSDUMPING</td>
<td>SYSDUMP or NOSYSDUMP</td>
</tr>
</tbody>
</table>
Table 29. Default values for system dump table entries (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Default</th>
<th>Attribute</th>
<th>Permitted value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take system dumps on related systems?</td>
<td>NO</td>
<td>DUMPSCOPE</td>
<td>LOCAL or RELATED</td>
</tr>
<tr>
<td>Shut down CICS?</td>
<td>NO</td>
<td>SHUTOPTION</td>
<td>SHUTDOWN or NOSHUTDOWN</td>
</tr>
<tr>
<td>Is dump eligible for DAE?</td>
<td>NO</td>
<td>DAEOPTION</td>
<td>DAE or NODAE</td>
</tr>
<tr>
<td>Maximum times dump code action can be taken</td>
<td>999</td>
<td>MAXIMUM</td>
<td>0 through 999</td>
</tr>
</tbody>
</table>

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 30 shows some examples of the sort of information that might be maintained in the transaction dump table for different transaction dump codes.

Table 30. Examples of transaction dump table entries

<table>
<thead>
<tr>
<th>Type of information</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction dump code</td>
<td>MYAB</td>
<td>ASRA</td>
<td>AKC3</td>
</tr>
<tr>
<td>Take a transaction dump?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Take a system dump?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Take system dumps on related systems?</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Shut down CICS?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Maximum times dump code action can be taken</td>
<td>50</td>
<td>30</td>
<td>999</td>
</tr>
<tr>
<td>Times dump code action already taken (current count)</td>
<td>0</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td>Transaction dumps taken</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Transaction dumps suppressed</td>
<td>0</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td>System dumps taken</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>System dumps suppressed</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
</tbody>
</table>
• **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 master terminal operator purges a task.

The entry shows that no action at all is to be taken in the event of such an abend. 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 31 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 31. Effect of global suppression of system dumping on transaction dump table update**

<table>
<thead>
<tr>
<th>Type of information</th>
<th>Before update</th>
<th>System dumping enabled</th>
<th>System dumping suppressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction dump code</td>
<td>MYAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Take a transaction dump?</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Take a system dump?</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Take system dumps on related systems?</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shut down CICS?</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum times action can be taken</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Times action already taken</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Transaction dumps taken</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Transaction dumps suppressed</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>System dumps taken</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 31. Effect of global suppression of system dumping on transaction dump table update (continued)

<table>
<thead>
<tr>
<th>Type of information</th>
<th>Before update</th>
<th>System dumping enabled</th>
<th>System dumping suppressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>System dumps suppressed</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The statistics show that a system dump was taken when system dumping was enabled, but not when system dumping was suppressed.

There is a further effect. CICS maintains a record of the current dump ID—the number of the most recent dump to be taken. This is printed at the start of the dump, together with the appropriate dump code. It is concatenated with the CICS run number—that is, the number of times that CICS has been brought up since the global catalog was created—to provide a unique ID for the dump.

**Note:** This does not apply to SDUMPs taken by the kernel; these always have a dump ID of 0/0000.

For example, for the ninth dump to be taken during the eleventh run of CICS, if the dump code were TD01, this is what you would see:

```
CODE=TD01 ID=11/0009
```

If system dumping is enabled for the dump code, the same dump ID is given to both the transaction dump and the system dump.

The system dump table

**Table 32** shows two examples of the sort of information that might be maintained in the system dump table for different system dump codes.

**Table 32. Examples of system dump table entries**

<table>
<thead>
<tr>
<th>Type of information</th>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>System dump code</td>
<td>SYDMP001</td>
<td>MT0001</td>
</tr>
<tr>
<td>Take a system dump?</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Take system dumps on related systems?</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Is the dump eligible for DAE?</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Shut down CICS?</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Maximum times action can be taken</td>
<td>(default)</td>
<td>999</td>
</tr>
<tr>
<td>Times action already taken</td>
<td>0</td>
<td>79</td>
</tr>
<tr>
<td>System dumps taken</td>
<td>0</td>
<td>79</td>
</tr>
<tr>
<td>System dumps suppressed</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The sort of information kept in the system dump table is similar to that kept in the transaction dump table (see **Table 30 on page 278**).

- **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 master terminal, with CEMT PERFORM DUMP or CEMT PERFORM SNAP. CICS is not shut down when a dump is taken for this dump code. Also, the value of 999 for "maximum times action can be taken" shows that an unlimited number of dumps can be taken for this dump code. The current count ("times action already taken") shows that to date, 79 dumps have been requested using CEMT.

---

**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. For example, 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. When prompted for dump parameters in response to the DUMP COMM command, enter the reply:

```
REPLY nn,STRLIST=(STRNAME=DFHCFLS_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=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.

For more information about the MVS DUMP command, see [OS/390 MVS System Commands, GC28-1781](os390-mvs-system-commands-gc28-1781).

**Notes:**

1. 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 [OS/390 MVS Setting Up a Sysplex, GC28-1779](os390-mvs-setting-up-a-sysplex-gc28-1779).
2. If system-managed structure duplexing is available, then both instances of the structure will be dumped.

---

**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. For example, enter the following command at the console:

```
DUMP COMM=(named_counter_poolname)
```
In response to the DUMP command, the system prompts you with a reply number for the dump options you want to specify. When prompted for dump parameters in response to the DUMP COMM command, 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.

For more information about the MVS DUMP command, see [OS/390 MVS System Commands, GC28-1781](https://www.ibm.com/support/knowledgecenter/SSEPGG_3.2.0/com.ibm.mvsch Jumping/ckmjump.cfm).

**Notes:**

1. 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 [OS/390 MVS Setting Up a Sysplex, GC28-1779](https://www.ibm.com/support/knowledgecenter/SSEPGG_3.2.0/com.ibm.mvsvse Jumping/.$CFS$/ckmjump.cfm).

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

---

**Dumping a shared temporary storage list structure**

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. When prompted for dump parameters in response to the DUMP COMM command, 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.

For more information about the MVS DUMP command, see OS/390 MVS System Commands, GC28-1781.

Notes:
1. 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 OS/390 MVS Setting Up a Sysplex, GC28-1779.
2. If system-managed structure duplexing is available, then both instances of the structure will be dumped.

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

Table 33. ZCQTRACE dump codes

<table>
<thead>
<tr>
<th>Module</th>
<th>Dump code</th>
<th>When invoked</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTRZCP</td>
<td>AZCQ</td>
<td>Installing terminal when termid = terminal ID</td>
</tr>
<tr>
<td>DFHTRZZP</td>
<td>AZQZ</td>
<td>Merging terminal with TYPETERM when termid = terminal ID</td>
</tr>
<tr>
<td>DFHTRZXP</td>
<td>AZQX</td>
<td>Installing connection when termid = connection ID</td>
</tr>
<tr>
<td>DFHTRZIP</td>
<td>AZQI</td>
<td>Installing sessions when termid = connection ID</td>
</tr>
<tr>
<td>DFHTRZPP</td>
<td>AZQP</td>
<td>When termid = pool terminal ID</td>
</tr>
<tr>
<td>DFHZCQIQ</td>
<td>AZQQ</td>
<td>Inquiring on resource when termid = resource ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(resource = terminal or connection)</td>
</tr>
<tr>
<td>DFHZCQIS</td>
<td>AZQS</td>
<td>Installing a resource when termid = resource ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(resource = terminal or connection), or when ZCQTRACE,AUTOINSTALL is specified.</td>
</tr>
</tbody>
</table>

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.
Chapter 18. 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" and "Formatting system dumps," respectively.

### Formatting transaction dumps

You can format transaction dumps offline using the CICS dump utility program, DFHDU640. Individual transaction dumps must be formatted in their entirety, but you can control which dumps are formatted from the dump data set. You can select dumps to be formatted as follows:

- Those taken in a specific period of time
- Those associated with a specific transaction identifier
- Those taken for a specific transaction dump code
- Those having specific dump IDs—that is, those for specific CICS runs and dump count values.

You can also use the SCAN option with the dump utility program, to get a list of the transaction dumps recorded on the specified dump data set.

For information about using DFHDU640 to format transaction dumps, see the CICS Operations and Utilities Guide.

### Formatting system dumps

You can process system dumps from the dump data set using an invocation of the interactive problem control system (IPCS).

In prior releases, 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 - DFHPD640 is the formatting routine you must define to IPCS when formatting CICS Transaction Server for z/OS, Version 3 Release 1 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, BLSECT, normally in SYS1.PARMLIB, has the following entry for CICS:

```plaintext
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:

```
//IPCSARM DD DSN=SYS1.PARMLIB,DISP=SHR For BLSCECT
// DD DSN=CICSTS31.CICS.SDFHPARM,DISP=SHR For DFHIPCSP
```

Figure 34 shows the release-specific entries specified in DFHIPCSP.

```c
/* ================================================================ */
EXIT EP(DFHPD212) VERB(CICS212) ABSTRACT(+
   'CICS Version 2 Release 1.2 analysis')
EXIT EP(DFHPD321) VERB(CICS321) ABSTRACT(+
   'CICS Version 3 Release 2.1 analysis')
EXIT EP(DFHPD330) VERB(CICS330) ABSTRACT(+
   'CICS Version 3 Release 3 analysis')
EXIT EP(DFHPD410) VERB(CICS410) ABSTRACT(+
   'CICS Version 4 Release 1 analysis')
EXIT EP(DFHPD510) VERB(CICS510) ABSTRACT(+
   'CICS Transaction Server for OS/390 Release 1 analysis')
EXIT EP(DFHPD520) VERB(CICS520) ABSTRACT(+
   'CICS Transaction Server for OS/390 Release 2 analysis')
EXIT EP(DFHPD530) VERB(CICS530) ABSTRACT(+
   'CICS Transaction Server for OS/390 Release 3 analysis')
EXIT EP(DFHPD610) VERB(CICS610) ABSTRACT(+
   'CICS Transaction Server for z/OS V2 R1 analysis')
EXIT EP(DFHPD620) VERB(CICS620) ABSTRACT(+
   'CICS Transaction Server for z/OS V2 R2 analysis')
EXIT EP(DFHPD630) VERB(CICS630) ABSTRACT(+
   'CICS Transaction Server for z/OS V2 R3 analysis')
EXIT EP(DFHPD640) VERB(CICS640) ABSTRACT(+
   'CICS Transaction Server for z/OS V3 R1 analysis')
/* ================================================================ */
```

Figure 34. 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 CICSnnn 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 CICS640 dump exit enables you to process a CICS Transaction Server for z/OS, Version 3 Release 1 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 CICS640 '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.

For guidance about using the formatting keywords and levels, see CICS Operations and Utilities Guide.

A summary of system dump formatting keywords and levels

The component keywords specify which areas of CICS you want the CICS640 exit to format dump data for, and the level number operand specifies the amount of data you want formatted.

If you omit all of the component keywords, and you do not specify DEF=0, the CICS dump exit formats dump data for all components.

The CICS dump component keywords, and the levels you can specify for each of them, are as follows:

AI [=0|2]]
   Autoinstall model manager.
   AI=0  Suppress formatting of AI control blocks.
   AI=2  Format AI control blocks.

AP [=0|1|2|3]]
   Application domain.
   AP=0  Suppress formatting of AP control blocks.
   AP=1  Format a summary of addresses of the AP control blocks for each active transaction.
   AP=2  Format the contents of the AP control blocks for each active transaction.
   AP=3  Format level-1 and level-2 data.

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.

   Note: You must use angled brackets around the specified parameter.

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]]
Autoinstall model manager.
CQ=0  suppresses formatting of console queue control blocks
CQ=1  formats the console queue summary
CQ=2  formats 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).

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|2}]
   The dump domain.
   DU=0  Suppress formatting of DU control blocks.
   DU=2  Format the DU anchor block.
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.
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.

II=[{0|1|2}]
The IIOP domain.
II=0  Format a summary of the II control blocks.
II=1  Format the contents of the II control blocks
II=2  Format level-0 and level-1 data.

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.

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

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.

MN={0|1|2|3}
    The monitoring domain.
    MN=0  Suppress formatting of monitoring domain control blocks.
    MN=1  Format the monitoring status and monitoring dictionary summary.
    MN=2  Format the anchor block and monitoring control table.
    MN=3  Format level-1 and level-2 data.

MRO={0|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.
    PCT=0  Suppress formatting of the transaction definitions.
    PCT=2  Format the transaction definitions.

PG={0|1|2|3}
    The program manager domain.
    PG=0  Suppress formatting of program manager domain control blocks.
    PG=1  Format the program manager summary.
PG=2  Format the program manager control blocks, including the anchor block, the LLEs, the PGWEs, the PPTEs, the PLCBs, and the HTBs.
PG=3  Format level-1 and level-2 data.

PR[{0|2}]  Partner resource management.
PR=0  Suppress formatting of PR areas.
PR=2  Format the PR static storage and the partner resource table.

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.

RM[{0|2}]  The recovery manager domain.
RM=0  Suppress formatting of RM control blocks.
RM=2  Format RM control blocks.

RX[{0|2|3}]  The recoverable resource management 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), the bound, pending and committed SH request (SHREQUES) summaries.
SH=2  Format the anchor block (SHDMANC), SH unit of work pending queue, (SHPQUEUE), the bound, pending and committed SH requests (SHREQUES).
SH=3  Format level-1 and level-2 data.

SJ[{0|1|2}]  The JVM domain.
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
Supress formatting of storage manager domain control blocks.
SM=1
Format the dynamic storage areas (DSAs), task and domain storage subpools, transaction areas (SMXs), suspend queue, and subspace area (SUA) summaries.
SM=2
Format the anchor block (SMA), subpool control areas (SCAs), pagepool areas (PPAs), pagepool extent areas (PPXs), storage manager transaction areas (SMXs), subspace areas (SUAs), suspend queue elements (SQEs), page allocation maps (PAMs), DSA extent descriptors (DXEs), and DSA extent getmain descriptors (DXGs).
SM=3
Format level-1 and level-2 data.

SSA=0
Suppress formatting of the static storage areas address list.
SSA=2
Format the static storage areas address list.

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.

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
Suppress formatting of the FEPI static area.
SZ=1
Format the FEPI static area. Nothing is printed unless you have installed FEPI. See CICS Front End Programming Interface User's Guide for further information on this feature.

TCP=0
Suppress formatting of TCP control blocks.
TCP=1
Format the terminal control summary.
TCP=2
Format the terminal control table, the terminal input/output areas, and the AIDs.
TCP=3
Format level-1 and level-2 data.

TDP=0
Supress formatting of transient data control blocks.
TDP=1
Format the summary of transient data destinations.

TDP=2
Format the transient data static storage areas and the multiple strings control blocks (MRCBs).

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

TI={0|1|2|3}
The timer domain.
TI=0 Suppress formatting of timer domain control blocks.
TI=1 Format outstanding request details.
TI=2 Format the anchor block.
TI=3 Format level-1 and level-2 data.

TMP={0|2}
The table manager program.
TMP=0 Suppress formatting of table manager static storage and control blocks.
TMP=2 Format table manager static storage and control blocks.

TR={0|1|2|3}
The trace domain.
TR=0 Suppress formatting of trace.
TR=1 Format abbreviated trace.
TR=2 Format full trace.
TR=3 Format level-1 and level-2 data.

TRS={trace selectivity parameter(s)}
Trace entry selectivity.
This keyword is effective only if the TR keyword value is 1, 2, or 3.
The TRS component keyword allows you to exercise much the same selectivity over the formatting and printing of trace entries written in a trace internal to a system dump, as you may exercise over the formatting and printing of trace entries in an auxiliary trace.
The trace selectivity parameter may be any valid trace selectivity parameter available to DFHTU640 for the formatting of CICS auxiliary trace entries except the parameters PAGESIZE, ABBREV, and FULL. You may, as with DFHTU640, select any number of parameters from those available.
Note, however, that you must use angled brackets around the parameter, or sequence of parameters, that you specify. The format and default values of parameters used to select trace entries from an internal SDUMP trace, are the same as those that apply when you use DFHTU640 to format auxiliary trace entries.

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 interface.
WB=0
   Suppress formatting of web interface control blocks.
WB=1
   Format the web interface summary. This displays the current state of the CICS web interface, followed by a summary of the state blocks controlled by the state manager.
WB=2
   Format the control blocks. This displays the current state of the CICS web interface, followed by the web anchor block, the global work area and associated control blocks, and the web state manager control blocks.

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 Appendix A, “SDUMP contents and IPCS CICS VERBEXIT keywords,” on page 321.

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:

<table>
<thead>
<tr>
<th>Level</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>1</td>
<td>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.)</td>
</tr>
<tr>
<td>2</td>
<td>For those components not included in a specified list of keywords, format the control block information only.</td>
</tr>
<tr>
<td>3</td>
<td>For those components not included in a specified list of keywords, format the control block information and also (where applicable) the summary information.</td>
</tr>
</tbody>
</table>

Effects of omitting the DEF parameter

- If you omit the DEF parameter and do not specify any component keywords, the result is as if you specified DEF=3.
- If you omit the DEF parameter and specify one or more component keywords, the result is as if you specified DEF=0.

Exceptions to the scope of the DEF parameter

There are two parts of a CICS system dump that are not governed by component keywords, and are therefore outside the scope of the DEF parameter. These are:
1. The dump summary
2. The error message index.

These parts of a CICS system dump are always formatted, even if you specify DEF=0 and no component keywords.

Interpreting transaction dumps

This section describes the contents of a transaction dump, and gives guidance on locating information that is useful for debugging transactions. The different parts of the transaction dump are dealt with in the order in which they appear, but be aware that only those parts that 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.
Job log

The job log for the dump utility program, DFHDX640, is sometimes shown at the start of the transaction dump, depending on how the job was invoked. You can ignore it, because it does not contain any information that can be used for debugging.

Symptom string

The symptom string tells you something about the circumstances of the transaction dump. It might show, for example, that the dump was taken because the transaction abended with abend code ASRA.

If you refer the problem that caused the dump to be taken to the IBM Support Center, they can use the symptom string to search the RETAIN database for problems that resemble it.

CICS Transaction Server for z/OS level

The CICS Transaction Server for z/OS level shows you what level of CICS Transaction Server for z/OS was being executed when the transaction dump was taken.

Transaction environment summary

The transaction environment summary is a formatted summary of the transaction environment at the time the dump is taken.

Remote abend indicator

If the transaction abended remotely (the abend originally occurred in a remote distributed program link (DPL) server program), and the abend is being re-issued on the local system, a message indicates this. The message contains the SYSID of the system that passed the abend to the local system. This information is taken from the transaction abend control block (see "Transaction storage" on page 299).

PSW

If the transaction dump was taken in response to a local abend with abend code AICA, ASRA, ASRB, or ASRD, a PSW is formatted from the dump data set. It belongs to the program that was being executed when the abend occurred. It is taken from the transaction abend control block (see "Transaction storage" on page 299).

Registers at the time of interrupt

If the transaction abended locally with abend code AICA, ASRA, ASRB, or ASRD, you see a set of registers that belong to the program that was executing when the error was detected. They are taken from the transaction abend control block (see "Transaction storage" on page 299).

Execution key

If the transaction abended locally with abend code ASRA or ASRB, the execution key that is in force at the time of the abend is formatted. It is taken from the transaction abend control block (see "Transaction storage" on page 299).
Space

If the transaction abended locally with an ASRA or ASRB abend code, 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.

Registers at last EXEC command

If the transaction has issued any EXEC commands, then a set of registers is displayed. These are the registers from the last EXEC command that was issued.

Task control area (TCA)

The next thing in the transaction dump is the entire TCA. This contains information about the transaction to which the dump relates. Note that the user area precedes the system area.

Task control area, system area

The system area of the task control area is formatted separately, 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.

Transaction work area (TWA)

Any transaction work area relating to the transaction is formatted, if present.

EXEC interface structure (EIS)

The EIS contains information about the transaction and program specific to the execution interface component of CICS.

System EXEC interface block (SYSEIB)

This is used solely by programs using the SYSEIB option. See CICS Application Programming Guide for more details.

EXEC interface user structure (EIUS)

The EIUS contains execution interface component information that must reside in user key storage.

EXEC interface block (DFHEIB)

DFHEIB contains information relating to the passing of EXEC requests from the program to CICS, and the passing of data between the program and CICS. Field EIBFN is of particular interest, because it shows the type of the last EXEC command to be issued by the program. For programming information about the values that EIBFN can contain, see CICS Application Programming Reference.

Kernel stack entries

The kernel stack entries contain information that has been saved by the kernel on behalf of programs and subroutines on the kernel linkage stack. If you refer the problem to the IBM Support Center, they might need to use the stack entries to find the cause of the failure.

Common system area (CSA)

The CSA is one of the main control areas used by CICS. It contains information relating to the system as a whole, and to the task that was running when the
transaction dump was invoked. It can be very useful for debugging both application problems and system problems. You cannot access fields in the CSA in your programs. Attempting to do so causes your transaction to terminate abnormally with abend code ASRD.

**Common system area optional features list (CSAOPFL)**

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

**Trace table (abbreviated format)**

The abbreviated-format trace table is formatted by default. You can suppress it by specifying the NOABBREV parameter in the DFHDU640 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 300.

**Trace table (extended format)**

Following the abbreviated-format trace table is the corresponding extended-format trace table. This is formatted by default. You can suppress it by specifying the NOFULL parameter in the DFHDU640 job. It contains more detail, but you can probably get all the information you need using the abbreviated trace.

**Common work area (CWA)**

The CWA is the installation-defined work area for use by all programs and is formatted if it exists.

**Transaction storage**

“Transaction storage” is storage that might have been obtained by CICS to store information about a transaction, or it might have been explicitly GETMAINed by the transaction for its own purposes. You are likely to find several such areas in the dump, each introduced by a header describing it as transaction storage of a particular class, for example:

- USER24
- USER31
- CICS24
- CICS31

Transaction storage class CICS31 contains, among other things, the transaction abend control block (TACB). To find it, look for the eye-catcher DFHTACB. DFHTACB contains valuable information relating to the abend. It contains:

- The PSW and general purpose registers of the program executing at the time of the abend (for local AICA, ASRA,ASRB and ASRD abends only. However, for some AICA abends, only the “next sequential instruction” part of the PSW and the registers are present.) Registers 12 and 13 contain the addresses of the TCA and CSA respectively in all abends except for ASRA and ASRB abends, when these registers contain data as at the time of the abend.
- The name of the failing program.
- The offset within the failing program at which the abend occurred (for local ASRA, ASRB and ASRD abends only).
- The execution key at the time of the abend (for local ASRA and ASRB abends only).
Whether the abend was caused by an attempt to overwrite a protected CICS DSA (local ASRA abends only).

Whether the program is executing in a subspace or the basespace.

The subspace STOKEN.

The subspace’s associated ALET.

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

**Terminal control table terminal entry (TCTTE)**

The TCTTE contains information about the terminal that is the principal facility of the transaction. You usually find one TCTTE for the transaction if the transaction is terminal oriented. For “daisy chained” transactions, you may find more than one.

To look at the TIOA for the task, find the address in field TCTTEDA of the TCTTE.

**Program information for the current transaction**

This shows summary information for all linked programs for the transaction that have not yet returned, including load point, entry point, and length. This is followed by the program storage for each of these programs. This is where you can find the instructions addressed by register 14 and by the PSW, and hence the point of failure in your program. For details of how you do this, see “Locating the last command or statement.”

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

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.

**Module index**

The final item that you find in the transaction dump is the module index. This shows you all the modules that were in storage when the error was detected, and their addresses.

**Locating the last command or statement**

The easiest way of locating the last command issued by your application before the abend is to use the internal trace table. You must have had internal trace running, and you need to have captured entries from the EI level-1 and PC level-1 trace points for your task. The way you can find the last command using trace is described in “Locating the last command or statement.”

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 301.
Last command identification

This process for identifying the last command applies to system dumps, in which it is necessary to identify the abending task. Item 2 is relevant to transaction dumps.

1. If the abend was a local AICA, ASRA, ASRB, or ASRD, find the last entry in the table and work back until you find an entry for trace point ID AP 1942. If you cannot find AP 1942, then search for any one of the following: AP 0790, AP 0791, or AP 0792. The trace entry for AP 1942 is made on entry to APLI’s recovery routine. The entries for AP 0790, AP 0791, and AP 0792 are made by DFHSRP, the AP domain recovery routine that deals with program checks, operating system abends, and runaway task conditions. The task number for your task is shown in the entry.

   If the abend was none of those mentioned above, find the last entry in the table and work back until you find an entry for trace point ID AP 00F2 (PCP abend) that references the abend code. The task number of your task is shown in the entry.

2. Now go back until you find the last trace entry showing your task number that was made from trace point ID AP 00E1. The trace entry is located in the EXEC interface program, DFHEIP. The data in the trace entry includes the value of EIBFN, which tells you the specific command your program issued before the abend. For programming information about the possible values that EIBFN can take, and their meanings, see [CICS Application Programming Reference](#).

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 CICS31 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. You should be able to see the eye-catcher **DFHTACB** in at least one of the areas. This signifies the start of the transaction abend control block, and it contains the registers and PSW belonging to the program being executed when the abend occurred. If there is more than one area containing this eye-catcher, it means that two or more successive abends occurred. You need to 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 should show a return address into your program. Either way, you need to correlate the address with a statement in the source code for the program.

If the source language is assembler, the instruction where the abend occurred is apparent from the program storage in the dump. If the source language is COBOL, PL/I, or C, you need to refer to a compiler output mapping source statements onto the object code.

**Locating program data**

You might want to look at the data that your application program has in its storage areas. CICS maintains a pointer to the chain of dynamic storage that the program uses, in field TCAPCDSA of the system area of the TCA.

- For PL/I programs, TCAPCDSA addresses the chain of PL/I DSAs.
- For COBOL programs, TCAPCDSA addresses the task global table (TGT) and working storage.
- For assembler programs, TCAPCDSA addresses the DFHEISTG storage.

You need to look in the appropriate programming language manual for details of the structure of the program’s acquired storage.

CICS does not support 64-bit addressing execution, but programs can use storage at addresses which are only available when CICS is running on 64-bit architecture machines. The CICS dump formatter displays the contents of the 64-bit General Purpose Registers captured when an abend occurs.

**Dumps for C/370 programs**

If DUMP(YES) is coded on the transaction definition, CICS system and transaction dumps are produced for failing C/370™ programs. The use of the relevant C/370 registers is as follows:

<table>
<thead>
<tr>
<th>Register</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>In most circumstances, is the base register</td>
</tr>
<tr>
<td>12</td>
<td>Holds the address of the CICS TCA for the C/370 program</td>
</tr>
<tr>
<td>13</td>
<td>Holds the address of the register save area</td>
</tr>
</tbody>
</table>

**Location of COBOL dumped areas**

The dumped COBOL program can be found 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 is addressed by TCAPCDSA (alias TCAPCCA) in the system part of the TCA (and forms part of the transaction storage chain). The COBOL area contains:

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

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 program's current registers 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 [CICS Supplied Transactions](#).

Formatting a coupling facility data table pool dump

To format a coupling facility data table structure dump, use the IPCS STRDATA subcommand. 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.

You can display the table data by converting 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.

For information about the STRDATA subcommand and its options, see OS/390 MVS IPCS Commands, GC28-1754.

### Formatting a named counter pool dump

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

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

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.

For information about the STRDATA subcommand and its options, see OS/390 MVS IPCS Commands, GC28-1754.

### Formatting a shared temporary storage pool dump

To format a shared temporary storage pool dump, use the IPCS STRDATA subcommand. 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.

You can display the queue data for a large queue by converting the data list number to decimal and specifying it on another STRDATA subcommand, so 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.

For information about the STRDATA subcommand and its options, see OS/390 MVS IPCS Commands, GC28-1754.
Chapter 19. The global trap/trace exit

The global trap/trace exit (DFHTRAP) is intended to be used only under the guidance of IBM Service personnel. It is designed so that a detailed diagnosis of a problem can be made without having to stop and then restart CICS.

Typically, the global trap/trace exit is used to detect errors that cannot be diagnosed by other methods. These might cause intermittent problems that are difficult to reproduce, the error perhaps occurring some time before the effects are noticed. For example, a field might be changed to a bad value, or some structure in storage might be overlaid at a specific offset.

DFHTRAP is an assembler language program that can be invoked whenever the trace (TR) domain is called to make a trace entry. The trap must be activated before it is used, either dynamically or at CICS initialization.

A skeleton version of DFHTRAP is supplied in both source and load-module forms. The source of DFHTRAP is cataloged in the CICS640.SDFHMAC library. The source of the skeleton DFHTRAP contains comments explaining its use of registers and DSECTs, and the coding you need to do if you need to use the exit.

The code in DFHTRAP must not make use of any CICS services, cause the current task to lose control, or change the status of the CICS system.

Establishing the exit

DFHTRAP can be installed as part of the RDO group DFHFE. You can install it either with the GRPLIST system initialization parameter, or dynamically using the CEDA transaction.

When CICS is running, activation and deactivation of the trap exit routine can be requested using the TRAP operand of the CSFE DEBUG command:

```
CSFE DEBUG,TRAP={ON|OFF}
```

The trap exit can also be activated at CICS initialization by specifying TRAP=ON, either in DFHSIT or as a startup override. If you want to replace a trap exit routine while CICS is running, use the CSFE DEBUG commands in conjunction with the CEMT NEWCOPY command. The following sequence of commands causes the currently active version of DFHTRAP to be refreshed:

```
CSFE DEBUG,TRAP=OFF
CEMT SET PROGRAM(DFHTRAP) NEWCOPY
CSFE DEBUG,TRAP=ON
```

Information passed to the exit

To assist in the diagnosis of faults in a CICS system, the trace domain passes information to the exit in a parameter list addressed by register 1. A DSECT (DFHTRADS) is supplied for this list, which contains the addresses of:

- The return-action flag byte
- The trace entry that has just been added to the table
- Up to three areas to be included in a further trace entry
- An 80-byte work area for the sole use of the trap exit
- The CSA (which may be zero during early initialization)
- The TCA, if there is one
• A register save area.

The DSECT also contains EQU statements for use in setting the return-action flag byte.

The exit can look at data from the current trace entry to determine whether or not the problem under investigation has appeared. It can also look at the TCA of the current task, and the CSA. The DSECTs for these areas are included in the skeleton source.

The CSA address is zero for invocations of DFHTRAP early in initialization, before the CSA is acquired.

---

**Actions the exit can take**

The return-action flag byte can be set by the exit to tell the trace domain what action is required on return from the exit. The following list gives the possible choices:

- Do nothing.
- Make a further trace entry.
- Take a CICS system dump using dump code TR1003.
- Terminate CICS without a dump after message DFHTR1000. (If you require a dump, the system dump return-action flag must also be set.)
- Disable the trap exit, so that it is not invoked again until reactivated by the CSFE transaction.

Any combination of these actions can be chosen and all actions are honored on return to the trace domain.

To reactivate the trap exit when it has been disabled, use CSFE DEBUG,TRAP=ON, unless the exit routine is to be replaced. In this case the sequence of commands given above applies.

The skeleton program shows how to make a further trace entry. When DFHTRAP detects a TS GET request, it asks for a further trace entry to be made by entering the data required in the area supplied for this purpose, and by setting the appropriate bit in the return-action flag byte.

The trace domain then makes a trace entry with trace point ID TR 0103, incorporating the information supplied by the exit.

Trace entries created in this way are written to any currently active trace destination. This could be the internal trace table, the auxiliary trace data set, or the GTF trace data set.

The skeleton DFHTRAP also shows how to detect the trace entry made by the storage manager (SM) domain for a GETMAIN request for a particular subpool. This is provided as an example of how to look at the data fields within the entry.
Program check handling

The occurrence of a program check in the trap exit is detected by the recovery routine in DFHTRPT. This then:

1. Marks the exit as unusable
2. Issues the message DFHTR1001 to the system console
3. Takes a CICS system dump with dump code TR1001, showing the PSW and registers at the time of the interrupt
4. Continues (ignoring the exit on future invocations of the trace domain).

To recover from this situation, execute the commands given above for replacing the current version of the exit routine.

Coding the exit

When using the trap exit, note the following points:

- A skeleton version of DFHTRAP is supplied in both source and load-module forms. Make sure that the library search sequence in the CICS startup JCL finds the correct version of the load module.

- DFHTRAP must save and restore the trace domain's registers. The supplied skeleton version contains the code necessary to do this. You are strongly advised not to change this code.

- The 80-byte work area provided for the sole use of the exit is in working storage acquired by the trace domain using an MVS GETMAIN. It is acquired and initialized to binary zeros when the trap is activated. It then exists until the trap is deactivated (CSFE DEBUG, TRAP=OFF). In a CICS transaction dump, the DFHTRAP working storage can be found soon after the CSA optional features list. The 80-byte work area is at the end of the DFHTRAP working storage and is immediately preceded by a 16-byte eye-catcher (DFHTRAP_WORKAREA), so that the work area can be located even if it has not been formatted. In a CICS system dump, the DFHTRAP working storage is in the trace domain (TR) section. See "Formatting system dumps" on page 285 for details of how to use the TR keyword to format the trace domain information in the dump.

- DFHTRAP must always run with AMODE(31) and RMODE(ANY) specified. In particular, it must always return control to the trace domain in 31-bit mode.
Part 4. Working with IBM to solve your problem

This part contains:

- Chapter 20, “IBM program support,” on page 311
- Chapter 21, “APARs, fixes, and PTFs,” on page 315
Chapter 20. IBM program support

The IBM Customer Engineering Program Support structure exists to help you resolve problems with IBM products, and to ensure that you can make the best use of your IBM computing systems. Program support is available to all licensed users of IBM licensed programs, and you can get assistance by telephoning your local Support Center.

This section helps you decide when to contact the Support Center, and what information you need to collect before contacting the Center. The section also gives you an understanding of the way in which IBM Program Support works.

When to contact the Support Center

Before contacting the Support Center, try to ensure that the problem belongs with the Center. Do not worry if you cannot be sure that the problem is due to CICS itself. How sure you are depends on the complexity of your installation, the experience and skill levels of your systems staff, and the symptoms that you have been getting.

In practice, many errors reported to Program Support turn out to be user errors, or they cannot be reproduced, or they need to be dealt with by other parts of IBM Service. This indicates just how difficult it can be to determine the precise cause of a problem. User errors are mainly caused by faults in application programs and errors in setting up systems. TCT parameters, in particular, have been found to cause difficulty in this respect.

Dealing with the Support Center

Your first contact with the Support Center is the call receipt operator, who takes initial details and routes your call to the correct support group.

The Support Center needs to know as much as possible about your problem, and you should have the information ready before making your first call. It is a good idea to put the information down on a problem reporting sheet, such as this one:
There are two advantages of using a problem reporting sheet:

1. You will be communicating with the IBM Support Center by telephone. So, with all your findings before you on a sheet of paper, you are prepared to respond to the questions that you may be asked.

2. You should maintain your own in-house tracking system to record all problems. This information can then be used for planning, organizing, communicating, and establishing priorities for controlling and solving these problems.

What the Support Center needs to know

When you contact the Support Center, you need to give the operator the name of your organization and your access code. Your access code is a unique code authorizing you to use IBM Software Services, and you provide it every time you contact the Center. Using this information, the operator accesses your customer profile, which contains details of your address, relevant contact names, telephone numbers, and details of the IBM products at your installation.

The Support Center operator asks you if this is a new problem, or a further call on an existing one. If it is new, you are assigned a unique incident number. A problem management record (PMR) is opened on the RETAIN system, where all activity associated with your problem is recorded. The problem remains “open” until it is solved.

Make a note of the incident number on your own problem reporting sheet. The Center expects you to quote the incident number in all future calls connected with this problem.

If the problem is new to you, the operator asks you for the source of the problem within your system software—that is, the program that seems to be the cause of the
problem. As you are reading this book, it is likely that you have already identified CICS as the problem source. You also need to give the version and release number, for example Version 4 Release 1.

You need to give a severity level for the problem. Severity levels can be 1, 2, or 3. They have the following meanings:

- Severity level 1 indicates that you are unable to use a program, resulting in a critical condition that needs immediate attention.
- Severity level 2 indicates that you are able to use the program, but that operation is severely restricted.
- Severity level 3 indicates that you are able to use the program, with limited functions, but the problem is not critical to your overall operation.

When deciding the severity of the problem, take care neither to understate it nor to overstate it. The Support Center procedures depend on the severity level so that the most appropriate use can be made of the Center’s skills and resources. Your problem is normally dealt with immediately if it is severity level 1.

Finally, the call receipt operator offers you a selection of specific component areas within CICS (for example, terminal control, file control) and asks you to choose the area where your problem appears to lie. Based on this selection, the operator can route your call to a specialist in the chosen area.

The keywords are subsequently used as search arguments on the RETAIN database, to see if your problem is a known one that has already been the subject of an authorized program analysis report (APAR).

You are not asked for any more information at this stage. However, you need to keep all the information relevant to the problem, and any available documentation such as dumps, traces, and translator, compiler, and program output.

How your problem is subsequently progressed depends on its nature. The representative who handles the problem gives you guidance about what is required from you. The possibilities are described in the next section.

**What happens next**

Details of your call are passed using the RETAIN problem management system to the appropriate support group. Your problem, assuming it is one associated with CICS, is put on the CICS queue. The problems are dealt with in order of severity level.

At first, a support center representative uses the keywords that you have provided to search the RETAIN database. If your problem is found to be one already known to IBM, and a fix has been devised for it, a Program Temporary Fix (PTF) can quickly be dispatched to you.

Let the representative know if any of the following events occurred before the problem appeared:

- Changes in level of MVS or licensed programs
- Regeneration of any product
- PTFs applied
- Additional features used
- Application programs changed
- Unusual operator action.
You might be asked to give values from a formatted dump or trace table. You might also be asked to carry out some special activity, for example to set a trap, or to use trace with a certain type of selectivity, and then to report on the results.

It might be necessary to have several follow-up telephone calls, depending on the complexity of the symptoms and your system environment. In every case, the actions taken by you and the Support Center are entered in the PMR. The representative can then be acquainted with the full history of the problem before any follow-up call.

The result of the investigations determines whether your problem is a new one, or one that is already known. If it is already known, and a fix has been developed, the fix is sent to you.

If the problem is new, an APAR may be submitted. This is dealt with by the CICS change team. See Chapter 21, “APARs, fixes, and PTFs,” on page 315.

**Reporting a FEPI problem to IBM**

For information specifically about reporting a FEPI problem, see the CICS Front End Programming Interface User’s Guide.
Chapter 21. APARs, fixes, and PTFs

An APAR is an “authorized program analysis report”. An APAR is your means of informing the appropriate change team of a problem you have found with an IBM program.

When the change team solves the problem, they produce a fix enabling you to get your system running properly again. Finally, a PTF is produced to replace the module in error, and the APAR is closed.

The APAR process

The first step in the APAR process is that a support center representative enters your APAR into the RETAIN system. The APAR text contains a description of your problem. If you have found a means of getting round the problem, details of this are entered as well. Your name is also entered, so that the Support Center knows who to contact if the change team need to ask anything further about the APAR documentation.

When the APAR is entered, you are given an APAR number. You must write this number on all the documentation you submit to the change team. This number is always associated with the APAR and its resolution and, if a code change is required, it is associated with the fix as well.

The next stage in the APAR process, getting relevant documentation to the change team, is up to you.

The following is a summary of the things you need to do:

1. You must collect all of the documentation that is required for the APAR. You are given guidance by the support center on precisely what you need to send. The documentation that is required varies depending on the problem area, but "Collecting the documentation for the APAR" gives you an idea of the material that you should supply.

2. You need to package all the documentation and send it to the change team. The procedure for this is given in "Sending the documentation to the change team" on page 316.

3. Lastly, you need to apply the PTF resulting from the APAR, possibly after testing the fix on your system. This is described in "Applying the fix" on page 317.

Collecting the documentation for the APAR

As a general rule, the documentation you need to submit for an APAR includes all the material you need yourself to perform problem determination. Some of the documentation is common to all CICS problems, and some is specific to particular types of problem.

Make sure the problem you have described can be seen in the documentation you send. If the problem has ambiguous symptoms, you need to reveal the sequence of events leading up to the failure. Tracing is valuable in this respect, but you might be able to provide details that trace cannot give. You are encouraged to annotate your documentation, if your annotation is legible and if it does not cover up vital information. You can highlight data in any hard copy you send, using transparent highlighting markers. You can also write notes in the margins, preferably using a red pen so that the notes are not overlooked.

© Copyright IBM Corp. 1997, 2010
Finally, note that if you send too little documentation, or if it is unreadable, the change team will return the APAR marked “insufficient documentation”. It is, therefore, worthwhile preparing your documentation carefully and sending everything relevant to the problem.

The general documentation is described in [General documentation needed for all problems with CICS.](#) However, these are only guidelines—you must find out from the support center precisely what documentation you need to send for your specific problem.

**General documentation needed for all problems with CICS**

The following is a list of the general documentation you might be asked to submit for an APAR:

- Any hard or soft copy illustrating the symptoms of the problem.
- A system dump of the CICS address space. Format the whole system dump if you plan to submit hard copy. Otherwise, you can just send the system dump data set on tape.
- A CICS trace. Auxiliary trace and GTF trace are best, but internal trace will do if you do not have either of these. If internal trace has been running, it is in the system dump in any case.
- Listings of CICS management modules, such as DFHKCP, that are not supplied as object code only. **These should be unnecessary if preassembled modules are used.**
- Relevant CICS tables. Again, these should be unnecessary if preassembled versions are used.
- Listings of relevant application programs.
- Console logs.
- CICS logs (for example, the CSMT log) wherever possible. These contain information that is often overlooked. They are particularly useful when VTAM is in use.
- JCL listings. These may appear on dumps, and need not be sent twice.
- A list of PTFs and APARs applied. The System Modification Program (SMP) CICS control data set (CDS) provides this information and should be sent.
- Details of any user modifications.

**Sending the documentation to the change team**

Follow the directions in APAR II02709 when gathering documentation to send to the change team. If you do not have access to the RETAIN database, ask your IBM representative to obtain a copy of APAR II02709 for you. If you cannot submit the documentation electronically, use an APAR box, which you can obtain from your local IBM branch. APAR boxes are clearly marked as such, and they have a panel where tracking information such as the APAR number can be written.

**Packing and mailing the APAR box**

Ship all your documentation and notes in one or more APAR boxes, making sure that the boxes are marked, for example, “1 of 2”, and so on, if you need to use more than one.

If you include any magnetic tapes, ensure that this is clearly indicated on the outside of the box. This lessens the chance of their being stored in magnetic fields strong enough to damage the data.
To make sure the documentation reaches the correct destination, that is, the CICS change team, the box should be marked:

SHIP TO CODE 5U6

You also need a mailing label with the address of the CICS change team on it.

When the change team receives the package, this is noted in your APAR record on the RETAIN system. The team then investigates the problem. Occasionally, they need to ask the Support Center to contact you for more documentation, perhaps specifying some trap you must apply before getting it.

When the problem is solved, a code is entered on RETAIN to close the APAR, and you are provided with a fix.

You can enquire any time at your Support Center on how your APAR is progressing, particularly if it is a problem of high severity.

---

**Applying the fix**

When the change team have found a fix for your problem, they might want you to test it on your system. If they do ask you to test the fix, you are normally given two weeks to do it and to provide them with the results. However, you can ask for an extension if you are unable to complete the testing in that time.

When the team is confident that the fix is satisfactory, the APAR is certified by the CICS development team and the APAR is closed. You receive notification when this happens.

**The APAR becomes a PTF**

If the solution involves a change to code in a CICS module that you can assemble, you are sent the code change right away. The change is later distributed as a PTF.

If you cannot assemble the module yourself, because it involves a part of CICS that is object serviced, you might be supplied with a ZAP or a TOTEST PTF.

If you want a PTF to resolve a specific problem, you can order it explicitly by its PTF number through the IBM Support Center. Otherwise, you can wait for the PTF to be sent out on the standard distribution tape.
Appendix A. SDUMP contents and IPCS CICS VERBEXIT keywords

The following two tables provide a cross-reference between the CICS control blocks contained in an SDUMP and their associated IPCS CICS VERBEXIT keyword.

The first table provides a list of IPCS CICS VERBEXIT keywords and the CICS control blocks that they display.

The second table provides a list of all CICS control blocks in an SDUMP, alphabetically, with their associated IPCS CICS VERBEXIT keyword.

Finding the control blocks from the keywords

AI keyword
- AITMTE (AITM entry)
- AITMSSA (AITM static storage)

AP keyword
- CICS24 (task storage—below 16MB, CICS key)
- CICS31 (task storage—above 16MB, 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)
- TD (user transient data)
- TS (user temporary storage)
- TCA (task control area—user)
- USER24 (task storage—below 16MB, user key)
- USER31 (task storage—above 16MB, user key)

APS keyword
- CICS24 (task storage - below 16MB, CICS key)
- CICS31 (task storage - above 16MB, 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)
- TD (user transient data)
- TS (user temporary storage)
- TCA (task control area - user)
• USER24 (task storage - below 16MB, user key)
• USER31 (task storage - above 16MB, user key)

Refer to the Language Environment for OS/390 & VM Debugging Guide and Run-Time Messages manual for details of LE control blocks and data areas.

BR keyword
• BRXA (bridge exit interface)
• bridge facility bitmap
• bridge facility keep chain
• TXN_CS
• bridge facility block

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)
• D2SS (CICS DB2 static storage)
• D2TRN (CICS DB2tran control block)

DD keyword
• ANCHOR (directory manager anchor block)
• DIR_HEAD (directory header)
• AVL_HEAD (AVL tree header)
• BRWS_VAL (browse value)
• HASH_TBL (hash table)
• HASHELEM (collision list element)

DH keyword
• DHA (document handler domain anchor block)
• DOA (document anchor block)
• DCR (document control record)
• DDB (document data block)
• DBB (document bookmark block)

DLI keyword
• CWE (CICS-DBCTL control work element)
• DFHDLI (CICS-DLI interface parameter list)
• DGB (CICS-DBCTL global block)
• DBCTA (DBCTL transaction area)
• DSF (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)

EJ keyword
- DFHEJANC (EJ domain anchor block)
- ejao (EJ domain object store anchor area)
- os_element (EJ object store elements)
- DFHEJANE (EJ domain elements anchor area)
- DFHEJCIE (CIE CorbaServer elements)
- DFHEJCBE (CBE CorbaServer browsers)
- DFHEJDIE (DIE DJar elements)
- DFHEJDBE (DBE DJar browsers)
- DFHEJBIE (BIE bean elements)
- DFHEJBBE (BBE bean browsers)
- EJDU_BLOCK (Task related Java diagnostics)
- SYSLIB (DFHEJDUB)

EM keyword
- EVA (event pool anchor)
- EMA (EM anchor block)
- EVB (event pool block)

FCP keyword
- ACB (VSAM ACBs)
- AFCCTE (application file control table element)
- DCB (data control block)
- DSNB (data set name block)
- DFHDTTABLE (data table base area)
- DFHDTFILE (data table path area)
- DFHDTHEADER (data table global area)
- DTRGLOBL (data table remote global area)
- FBWA (data table browse area)
- FCSTATIC (FCP static storage—anchor block)
- FCCTE (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.

II keyword
- II domain anchor block
- MDA (request model - model class anchor block)
- request receiver - per request receiver task:
  - CONN_DATA (connection data)
  - cicsTaskTrackingContext
  - Saved service contexts
  - TXN_DATA (transaction data)
  - Replies - per reply outstanding
    - LISTEN_DATA
    - cicsTaskTrackingContext

- Request - per request outstanding
  - ruei containing pointer and length of each element of the request
  - Each element of the request

- Request processor - per request processor task:
  - RP_DATA (request processor data)
  - For each outstanding request received:
    - cicsTaskTrackingContext
  - For each outstanding reply:
    - LISTEN_DATA
    - cicsTaskTrackingContext

KE keyword
- AFCB (CICS AFCB)
- AFCS (CICS AFCS)
- AFT (CICS AFT)
- AUTOSTCK (automatic storage stack entry)
- KCB (kernel anchor block)
- KERNSTCK (kernel linkage storage stack entry)
- KERRD (kernel error data)
- KTCB (KTCB table entry)
- DOH (domain table header)
- DOM (domain table entry)
- TAH (task table header)
- TAS (task table entry—TASENTRY)
- TCH (KTCB table header)

LD keyword
- APE (active program element)
- CPE (current program element)
- CSECTL (program CSECT List)
- LD_GLBL (loader domain global storage—anchor block)
- LDBE (loader domain browse element)
- LDWE (loader domain wait element)
- LLA (load list area)

LG keyword
- LGA (log domain anchor)
- LGGL (general log data)
- LGSD (stream data)
- LGJI (journal information)
- LGJMC (journalmodel content)
- LGBR (stream/journal/journalmodel browse)
• 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)

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)

MRO keyword
• 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)

PCP keyword
- PPTTE (program processing table entries)

PCT keyword
- PCTTE (program control table entries)

PG keyword
- PGA (program manager anchor)
- LLE (load list element)—can be system LLE or task LLE
- PGWE (program manager wait element)
- PPTE (program processing table element)
- PTA (program transaction area)
- CHCB (channel control block)
- CPCB (container pool control block)
- CRCB (container control block)
- CSCB (container segment block)
- PLCB (program manager program level control block)
- HTB (handle table)


PI keyword
- PIA (pipeline manager anchor block)
- PIH (pipeline element header block)
- PEB (pipeline element block)
- SNE (service handler element)
- HPE (header program element)
- WHB (Web service header block)
- WCB (Web service control block)
- WRB (Web service resource block)

RM keyword
- RMUW (recovery manager)
- RMLK (recovery manager link)
- RMCD (recovery manager client directory)
- RMCI (recovery manager client identity)
- RMDM (recovery manager domain anchor)
- RMLI (recovery manager loggable identity)
- RNNM (recovery manager logname)
- RMRO (recovery manager resource owner)
- RMSL (recovery manager system log)

RX keyword
• Active Unit of Recovery data (CICS key)
• In-resync Unit of Recovery data (CICS key)
• Active Unit of Recovery data (Key 0)
• In-resync Unit of Recovery data (Key 0)
• RX domain anchor block (CICS key)
• DFHRXSVC dynamic storage area
• RX domain anchor block (Key 0)

**RZ keyword**
• RZDM (RZ domain anchor block)
• RZRMB (rzrmb instance data)
• RZREQSTR (rz_reqstream instance data)
• RZTR (rztr instance data)

**SJ keyword**
• SJA (JVM domain anchor block)
• SJCCH (Shared class cache control block)
• SJTCB (J8 or J9 TCB control block)
• SJVMS (Master JVM, or JVMset, control block)

**SM keyword**
• CTN (cartesian tree node)
• DXE (DSA extent list element)
• DXG (DSA extent getmain description)
• DXH (DSA extent list header)
• MCA (SM macro-compatibility control area)
• PAM (page allocation map)
• PPA (page pool control area)
• PPX (page pool extent control area)
• QPF (quickcell page free element)
• QPH (quickcell page header)
• SAE (storage access table entry)
• SAT (storage access table)
• SCA (subpool control area)
• SCE (storage element descriptor)
• SCF (free storage descriptor)
• SMA (storage manager domain anchor block)
• SMSVCTRT (DFHSMSCVC trace table)
• SMX (transaction storage area)
• SQE (suspend queue element)
• STAB (storage manager statistics buffer)
• SUA (subspace area)

**SO keyword**
• SOA (SO domain anchor block)
• LTE (Listener table entry)
• STE (Session table entry)
• TDA (Tcpipservice anchor block)
• TDB (Tcpipservice control block)
• TBR (Tcpipservice browse block)

**SSA keyword**
• SSAL (static storage address list)
• SSA (static storage areas)

**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 (VTAM 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 (VTAM 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 (VTAM receive-any Input area)
- RPL (VTAM request parameter list—receive-any RPLs)
- SNEX (TCTTE signon extension)
- TACLE (terminal abnormal condition line entry)
- TCTENIB (NIB descriptor)
- TCTESBA (APPC send/receive buffer)
- TCFX (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)

TMP keyword
• TMSTATIC (table manager static storage)
• SKT (scatter tables)
• DIRSEG (directory segments)
• TM_LOCKS (read lock blocks)

TR keyword
• TRA (trace domain anchor block)
• TRDCB (auxiliary trace data set DCB)
• TRDECB (auxiliary trace data set DECB)

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)
Finding the keywords from the control blocks

<table>
<thead>
<tr>
<th>CICS Control Block</th>
<th>VERBEXIT Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA</td>
<td>TS auxiliary control area</td>
</tr>
<tr>
<td>ACB</td>
<td>VSAM ACBs</td>
</tr>
<tr>
<td>ACB</td>
<td>VTAM access method control block</td>
</tr>
<tr>
<td>ACB</td>
<td>TD VSAM ACB</td>
</tr>
<tr>
<td>AFCB</td>
<td>CICS AFCB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CICS Control Block</th>
<th>VERBEXIT Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFCS</td>
<td>CICS AFCS</td>
</tr>
<tr>
<td>AFCTE</td>
<td>application file control table element</td>
</tr>
<tr>
<td>AFT</td>
<td>CICS AFT</td>
</tr>
<tr>
<td>AID</td>
<td>automatic initiation descriptor</td>
</tr>
<tr>
<td>AITM</td>
<td>static storage</td>
</tr>
<tr>
<td>AITMTE</td>
<td>autoinstall terminal models</td>
</tr>
<tr>
<td>ANCHOR</td>
<td>directory manager anchor block</td>
</tr>
<tr>
<td>APE</td>
<td>active program element</td>
</tr>
<tr>
<td>AUTOSTCK</td>
<td>automatic storage stack entry</td>
</tr>
<tr>
<td>AVL_HEAD</td>
<td>AVL tree header</td>
</tr>
<tr>
<td>AWE</td>
<td>autoinstall work element</td>
</tr>
<tr>
<td>BCA</td>
<td>TS buffer control area</td>
</tr>
<tr>
<td>BIND</td>
<td>bind image</td>
</tr>
<tr>
<td>BITMAPn</td>
<td>instance of resource naming BITMAP</td>
</tr>
<tr>
<td>BMH</td>
<td>TS byte map header</td>
</tr>
<tr>
<td>BMP</td>
<td>TS byte map</td>
</tr>
<tr>
<td>BRB</td>
<td>TS browse block</td>
</tr>
<tr>
<td>BRWS_VAL</td>
<td>browse value</td>
</tr>
<tr>
<td>BRXA</td>
<td>bridge exit interface</td>
</tr>
<tr>
<td>BRXAS</td>
<td>bridge facility bitmap</td>
</tr>
<tr>
<td>BRXAK</td>
<td>bridge facility keep chain</td>
</tr>
<tr>
<td>BRXAKS</td>
<td>bridge facility block</td>
</tr>
<tr>
<td>TXN_CS</td>
<td>BR</td>
</tr>
<tr>
<td>BUFFER</td>
<td>TD I/O buffer</td>
</tr>
<tr>
<td>CAVM_STA</td>
<td>CAVM static storage</td>
</tr>
<tr>
<td>CC_ACB</td>
<td>local catalog ACB</td>
</tr>
<tr>
<td>CC_RPL</td>
<td>local catalog RPLs, one each thread</td>
</tr>
<tr>
<td>CCB</td>
<td>connection control block</td>
</tr>
<tr>
<td>CCBUFFER</td>
<td>local catalog buffers, one each thread</td>
</tr>
<tr>
<td>CCIN</td>
<td>CICS client CCIN parameters</td>
</tr>
<tr>
<td>CHCB</td>
<td>channel control block</td>
</tr>
<tr>
<td>CICSS24</td>
<td>task storage - below 16 MB, CICS key</td>
</tr>
<tr>
<td>CICSS31</td>
<td>task storage - above 16 MB, CICS key</td>
</tr>
<tr>
<td>CONN_DATA</td>
<td>connection data</td>
</tr>
<tr>
<td>CPCB</td>
<td>container pool control block</td>
</tr>
<tr>
<td>CPE</td>
<td>current program element</td>
</tr>
<tr>
<td>CPSTATIC</td>
<td>common program interface storage</td>
</tr>
<tr>
<td>CRB</td>
<td>CICS region block</td>
</tr>
<tr>
<td>CRCB</td>
<td>container control block</td>
</tr>
<tr>
<td>CSA</td>
<td>common system area</td>
</tr>
<tr>
<td>CSCB</td>
<td>container segment block</td>
</tr>
<tr>
<td>CTIN</td>
<td>CICS client CTIN parameters</td>
</tr>
<tr>
<td>CTN</td>
<td>cartesian tree node</td>
</tr>
<tr>
<td>CSAOPFL</td>
<td>CSA optional features list</td>
</tr>
<tr>
<td>CSB</td>
<td>connection status block</td>
</tr>
<tr>
<td>CSECTL</td>
<td>program CSECT list</td>
</tr>
<tr>
<td>CWA</td>
<td>common work area</td>
</tr>
<tr>
<td>CWE</td>
<td>CICS/DBCTL control work element</td>
</tr>
<tr>
<td>D2CSB</td>
<td>CICS DB2 subtask block</td>
</tr>
<tr>
<td>D2ENT</td>
<td>CICS DB2entry control block</td>
</tr>
<tr>
<td>CICS Control Block</td>
<td>VERBEXIT</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
</tr>
<tr>
<td>D2GLB</td>
<td>CICS DB2 global block</td>
</tr>
<tr>
<td>D2LOT</td>
<td>CICS DB2 life of task block</td>
</tr>
<tr>
<td>D2SS</td>
<td>CICS DB2 static storage</td>
</tr>
<tr>
<td>D2TRN</td>
<td>CICS DB2tran control block</td>
</tr>
<tr>
<td>DCB</td>
<td>data control block</td>
</tr>
<tr>
<td>DGB</td>
<td>BSAM data control block</td>
</tr>
<tr>
<td>DCTE</td>
<td>transient data queue definitions</td>
</tr>
<tr>
<td>DFHDLDP</td>
<td>CICS/DLI interface parameter list</td>
</tr>
<tr>
<td>DFHEJANC</td>
<td>EJ domain anchor block</td>
</tr>
<tr>
<td>DFHEJANE</td>
<td>EJ domain elements anchor area</td>
</tr>
<tr>
<td>DFHEJBBE</td>
<td>BBE bean browsers</td>
</tr>
<tr>
<td>DFHEJBIE</td>
<td>BIE bean elements</td>
</tr>
<tr>
<td>DFHEJCIE</td>
<td>CIE CorbaServer elements</td>
</tr>
<tr>
<td>DFHEJCBIE</td>
<td>CBE CorbaServer browsers</td>
</tr>
<tr>
<td>DFHEJDDBE</td>
<td>DBE DJAR browsers</td>
</tr>
<tr>
<td>DFHEJDIE</td>
<td>DIE DJAR elements</td>
</tr>
<tr>
<td>DFHSIT</td>
<td>system initialization table</td>
</tr>
<tr>
<td>DGBCTA</td>
<td>CICS/DBCTL global block</td>
</tr>
<tr>
<td>DIB</td>
<td>data interchange block</td>
</tr>
<tr>
<td>DIB</td>
<td>directory header</td>
</tr>
<tr>
<td>DIB</td>
<td>directory segments</td>
</tr>
<tr>
<td>DMANCHOR</td>
<td>domain manager anchor block</td>
</tr>
<tr>
<td>DOH</td>
<td>domain table header</td>
</tr>
<tr>
<td>DOM</td>
<td>domain table entry</td>
</tr>
<tr>
<td>DPA</td>
<td>DP domain anchor block</td>
</tr>
<tr>
<td>DPTA</td>
<td>DP domain task area</td>
</tr>
<tr>
<td>DS_TCB</td>
<td>TCB block</td>
</tr>
<tr>
<td>DSANC</td>
<td>dispatcher anchor block</td>
</tr>
<tr>
<td>DSB</td>
<td>CICS/DBCTL scheduling block</td>
</tr>
<tr>
<td>DSNB</td>
<td>data set name block</td>
</tr>
<tr>
<td>DFHDTTABLE</td>
<td>data table base area</td>
</tr>
<tr>
<td>DFHDTFILE</td>
<td>data table path area</td>
</tr>
<tr>
<td>DFHDTHEADER</td>
<td>data table global area</td>
</tr>
<tr>
<td>DTA</td>
<td>dispatcher task area</td>
</tr>
<tr>
<td>DTN</td>
<td>TS digital tree node</td>
</tr>
<tr>
<td>DTRGLOBL</td>
<td>data table remote global area</td>
</tr>
<tr>
<td>DUA</td>
<td>dump domain anchor block</td>
</tr>
<tr>
<td>DUBUFFER</td>
<td>transaction dump-data set buffer</td>
</tr>
<tr>
<td>DUMTCTTE</td>
<td>dummy TCTTE</td>
</tr>
<tr>
<td>DWE</td>
<td>user DWE storage</td>
</tr>
<tr>
<td>DXE</td>
<td>DSA extent list element</td>
</tr>
<tr>
<td>DXG</td>
<td>DSA extent getmain descriptor</td>
</tr>
<tr>
<td>DXH</td>
<td>DSA extent list header</td>
</tr>
<tr>
<td>DXPS</td>
<td>CICS-DL/I-XRF anchor block</td>
</tr>
<tr>
<td>EIB</td>
<td>EXEC interface block</td>
</tr>
<tr>
<td>EIS</td>
<td>EXEC interface structure</td>
</tr>
<tr>
<td>EIUUS</td>
<td>EXEC interface user structure</td>
</tr>
<tr>
<td>ejao</td>
<td>EJ domain object store anchor area</td>
</tr>
<tr>
<td>EJDU_BLOCK</td>
<td>task-related Java diagnostics</td>
</tr>
<tr>
<td>EPB</td>
<td>exit program blocks</td>
</tr>
<tr>
<td>EXLST</td>
<td>VTAM ACB exit list</td>
</tr>
<tr>
<td>CICS Control Block</td>
<td>VERBEXIT Keyword</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>FBWA</td>
<td>data table browse area</td>
</tr>
<tr>
<td>FCSTATIC</td>
<td>FCP</td>
</tr>
<tr>
<td>FCSTE</td>
<td>FCP</td>
</tr>
<tr>
<td>FILE</td>
<td>user file storage</td>
</tr>
<tr>
<td>FLAB</td>
<td>file lasting access block</td>
</tr>
<tr>
<td>FRAB</td>
<td>FCP</td>
</tr>
<tr>
<td>FREECHAI</td>
<td>LM domain freechain 1</td>
</tr>
<tr>
<td>FREECHAI</td>
<td>LM domain freechain 2</td>
</tr>
<tr>
<td>FREECHAI</td>
<td>LM domain freechain 3</td>
</tr>
<tr>
<td>FRTE</td>
<td>file request thread element</td>
</tr>
<tr>
<td>GC_ACB</td>
<td>global catalog ACB</td>
</tr>
<tr>
<td>GC_RPL</td>
<td>global catalog RPLs, one each thread</td>
</tr>
<tr>
<td>GCBUFFER</td>
<td>global catalog buffers, one each thread</td>
</tr>
<tr>
<td>GWA</td>
<td>EPB global work area</td>
</tr>
<tr>
<td>GWA</td>
<td>web global work area</td>
</tr>
<tr>
<td>HASH_TBL</td>
<td>hash table</td>
</tr>
<tr>
<td>HASHELEM</td>
<td>collision list element</td>
</tr>
<tr>
<td>HPE</td>
<td>Header program element</td>
</tr>
<tr>
<td>HTB</td>
<td>handle table</td>
</tr>
<tr>
<td>ICE</td>
<td>interval control elements/AIDs</td>
</tr>
<tr>
<td>ICE</td>
<td>TS interval control element</td>
</tr>
<tr>
<td>IECCB</td>
<td>IE domain client conversation block</td>
</tr>
<tr>
<td>IECSB</td>
<td>IE domain connection status block</td>
</tr>
<tr>
<td>ISORM</td>
<td>indirect system object resolution map</td>
</tr>
<tr>
<td>JCA</td>
<td>journal control area</td>
</tr>
<tr>
<td>KCB</td>
<td>kernel anchor block</td>
</tr>
<tr>
<td>KERNSTCK</td>
<td>kernel linkage storage stack entry</td>
</tr>
<tr>
<td>KERRD</td>
<td>kernel error data</td>
</tr>
<tr>
<td>KTCB</td>
<td>KTCB table entry</td>
</tr>
<tr>
<td>LACB</td>
<td>logon address control block</td>
</tr>
<tr>
<td>LCB</td>
<td>logon control block</td>
</tr>
<tr>
<td>LD_GLBL</td>
<td>loader domain global storage - anchor block</td>
</tr>
<tr>
<td>LDPE</td>
<td>loader domain browse element</td>
</tr>
<tr>
<td>LDWE</td>
<td>loader domain wait element</td>
</tr>
<tr>
<td>LGA</td>
<td>log domain anchor block</td>
</tr>
<tr>
<td>LGGL</td>
<td>general log data</td>
</tr>
<tr>
<td>LGSD</td>
<td>stream data</td>
</tr>
<tr>
<td>LGJI</td>
<td>journal information</td>
</tr>
<tr>
<td>LGJMC</td>
<td>journalmodel content</td>
</tr>
<tr>
<td>LGBR</td>
<td>stream/journal/journalmodel browse</td>
</tr>
<tr>
<td>LGUOW</td>
<td>log manager unit of work token</td>
</tr>
<tr>
<td>Block class data</td>
<td>LG</td>
</tr>
<tr>
<td>Block instance data</td>
<td>LG</td>
</tr>
<tr>
<td>BrowseableStream class data</td>
<td>LG</td>
</tr>
<tr>
<td>BrowseableStream instance data</td>
<td>LG</td>
</tr>
<tr>
<td>Chain class data</td>
<td>LG</td>
</tr>
<tr>
<td>CICS Control Block</td>
<td>VERBEXIT Keyword</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Chain instance data</td>
<td>LG</td>
</tr>
<tr>
<td>HardStream instance data</td>
<td>LG</td>
</tr>
<tr>
<td>L2 anchor block</td>
<td>LG</td>
</tr>
<tr>
<td>Stream class data</td>
<td>LG</td>
</tr>
<tr>
<td>Stream instance data</td>
<td>LG</td>
</tr>
<tr>
<td>SuspendQueue elements</td>
<td>LG</td>
</tr>
<tr>
<td>SystemLog class data</td>
<td>LG</td>
</tr>
<tr>
<td>load list element</td>
<td>PG</td>
</tr>
<tr>
<td>lock manager domain anchor block</td>
<td>LM</td>
</tr>
<tr>
<td>LM domain quickcell 1</td>
<td>LM</td>
</tr>
<tr>
<td>LM domain quickcell 2</td>
<td>LM</td>
</tr>
<tr>
<td>LM domain quickcell 3</td>
<td>LM</td>
</tr>
<tr>
<td>LM domain lock element</td>
<td>LM</td>
</tr>
<tr>
<td>extracted logon or CLSDST pass data</td>
<td>TCP</td>
</tr>
<tr>
<td>local userid table element</td>
<td>TCP</td>
</tr>
<tr>
<td>LX array</td>
<td>MRO</td>
</tr>
<tr>
<td>user BMS MAP storage</td>
<td>AP APS</td>
</tr>
<tr>
<td>TD buffer control area</td>
<td>TDP</td>
</tr>
<tr>
<td>TD buffer control block</td>
<td>TDP</td>
</tr>
<tr>
<td>SM macro-compatibility control area</td>
<td>SM</td>
</tr>
<tr>
<td>monitoring control table</td>
<td>MN</td>
</tr>
<tr>
<td>request model model class anchor block</td>
<td>II</td>
</tr>
<tr>
<td>message domain anchor block</td>
<td>ME</td>
</tr>
<tr>
<td>monitor domain global storage - anchor block</td>
<td>MN</td>
</tr>
<tr>
<td>monitor authorization facility parameter block</td>
<td>MN</td>
</tr>
<tr>
<td>monitor field connectors</td>
<td>MN</td>
</tr>
<tr>
<td>monitor dictionary</td>
<td>MN</td>
</tr>
<tr>
<td>exception record buffer</td>
<td>MN</td>
</tr>
<tr>
<td>user EMP address list</td>
<td>MN</td>
</tr>
<tr>
<td>excluded/included CICS field map</td>
<td>MN</td>
</tr>
<tr>
<td>performance data buffer</td>
<td>MN</td>
</tr>
<tr>
<td>SMF record buffer</td>
<td>MN</td>
</tr>
<tr>
<td>transaction monitoring area</td>
<td>MN</td>
</tr>
<tr>
<td>MVS WLM performance blocks</td>
<td>MN</td>
</tr>
<tr>
<td>TD queue control block</td>
<td>TDP</td>
</tr>
<tr>
<td>TD string control area</td>
<td>TDP</td>
</tr>
<tr>
<td>TD string control block</td>
<td>TDP</td>
</tr>
<tr>
<td>TD CI state map - segment descriptor</td>
<td>TDP</td>
</tr>
<tr>
<td>TD wait control block</td>
<td>TDP</td>
</tr>
<tr>
<td>XM domain MXT tclass</td>
<td>XM</td>
</tr>
<tr>
<td>node initialization block</td>
<td>TCP</td>
</tr>
<tr>
<td>persistent session INQUIRE PERSESS NI Bs</td>
<td>TCP</td>
</tr>
<tr>
<td>transaction dump open block</td>
<td>DU</td>
</tr>
<tr>
<td>EJ object store elements</td>
<td>EJ</td>
</tr>
<tr>
<td>OT domain anchor block</td>
<td>OT</td>
</tr>
<tr>
<td>override parameter temporary work area</td>
<td>PA</td>
</tr>
<tr>
<td>parameter manager domain anchor block</td>
<td>PA</td>
</tr>
<tr>
<td>page allocation map</td>
<td>SM</td>
</tr>
<tr>
<td>DLI/DRA architected parameter list</td>
<td>DLI</td>
</tr>
<tr>
<td>CICS Control Block</td>
<td>VERBEXIT Keyword</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>PARMSAVE</td>
<td>SIT override parameters</td>
</tr>
<tr>
<td>PCA</td>
<td>TS pool control area</td>
</tr>
<tr>
<td>PCTTE</td>
<td>program control table entries</td>
</tr>
<tr>
<td>PEB</td>
<td>Pipeline element block</td>
</tr>
<tr>
<td>PGA</td>
<td>program management anchor</td>
</tr>
<tr>
<td>PGWE</td>
<td>program management wait element</td>
</tr>
<tr>
<td>PIA</td>
<td>Pipeline manager anchor block</td>
</tr>
<tr>
<td>PIH</td>
<td>Pipeline element header block</td>
</tr>
<tr>
<td>PLCB</td>
<td>program management program control blk</td>
</tr>
<tr>
<td>PPA</td>
<td>pagepool control area</td>
</tr>
<tr>
<td>PPTE</td>
<td>program processing table element</td>
</tr>
<tr>
<td>PPTTE</td>
<td>program processing table entries</td>
</tr>
<tr>
<td>PPA</td>
<td>pagepool extent control area</td>
</tr>
<tr>
<td>PRSS</td>
<td>persistent sessions CV29, FMH5, BIS, and BID data</td>
</tr>
<tr>
<td>PRSTATIC</td>
<td>partner resource static area</td>
</tr>
<tr>
<td>PRTE</td>
<td>partner resource table entries</td>
</tr>
<tr>
<td>PRVMODS</td>
<td>SIT PRVMOD list</td>
</tr>
<tr>
<td>PS_BID</td>
<td>persistent sessions OPNDST RESTORE BID</td>
</tr>
<tr>
<td>PS_BIND</td>
<td>persistent sessions INQUIRE PERSESS BIND</td>
</tr>
<tr>
<td>PS_BIS</td>
<td>persistent sessions OPNDST RESTORE BIS</td>
</tr>
<tr>
<td>PS_CV29</td>
<td>persistent sessions OPNDST RESTORE control vector 29</td>
</tr>
<tr>
<td>PS_FM5</td>
<td>persistent sessions OPNDST RESTORE FMH5</td>
</tr>
<tr>
<td>PS_MODN</td>
<td>persistent sessions INQUIRE PERSESS modename</td>
</tr>
<tr>
<td>PS_NIB</td>
<td>persistent sessions INQUIRE PERSESS NIB</td>
</tr>
<tr>
<td>PS_PLST</td>
<td>persistent sessions INQUIRE PERSESS parameter list</td>
</tr>
<tr>
<td>PS_POOL</td>
<td>persistent sessions RPL pool header</td>
</tr>
<tr>
<td>PS_RPL</td>
<td>persistent sessions RPL</td>
</tr>
<tr>
<td>PS_SESS</td>
<td>persistent sessions INQUIRE PERSESS session ID</td>
</tr>
<tr>
<td>PTA</td>
<td>program transaction area</td>
</tr>
<tr>
<td>PWE</td>
<td>postponed work element</td>
</tr>
<tr>
<td>QAB</td>
<td>TS queue anchor block</td>
</tr>
<tr>
<td>QOB</td>
<td>TS queue ownership block</td>
</tr>
<tr>
<td>QPH</td>
<td>quickcell page header</td>
</tr>
<tr>
<td>QPK</td>
<td>quickcell page free element</td>
</tr>
<tr>
<td>QUB</td>
<td>TS queue update block</td>
</tr>
<tr>
<td>RACE</td>
<td>receive-any control elements</td>
</tr>
<tr>
<td>RAIA</td>
<td>VTAM receive-any input area</td>
</tr>
<tr>
<td>RMCBS</td>
<td>recovery manager control blocks</td>
</tr>
<tr>
<td>RP_DATA</td>
<td>request processor data</td>
</tr>
<tr>
<td>RPL</td>
<td>VTAM request parameter list - receive-any RPLs</td>
</tr>
<tr>
<td>RPL TD VSAM RPL</td>
<td>TCP</td>
</tr>
<tr>
<td>RSB</td>
<td>remote scheduling block</td>
</tr>
<tr>
<td>RZDM</td>
<td>RZ domain anchor block</td>
</tr>
<tr>
<td>RZRM</td>
<td>rzmb instance data</td>
</tr>
<tr>
<td>RZREQSTR</td>
<td>rz Reqstream instance data</td>
</tr>
<tr>
<td>RZTR</td>
<td>rztr instance data</td>
</tr>
<tr>
<td>SAE</td>
<td>storage access table entry</td>
</tr>
<tr>
<td>SAT</td>
<td>storage access table</td>
</tr>
<tr>
<td>SBB</td>
<td>TS shared browse block</td>
</tr>
<tr>
<td>SCA</td>
<td>subpool control areas</td>
</tr>
</tbody>
</table>

Appendix A. SDUMP contents and IPCS CICS VERBEXIT keywords
<table>
<thead>
<tr>
<th>CICS Control Block</th>
<th>VERBEXIT Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCACB</td>
<td>subsystem connection address control block MRO</td>
</tr>
<tr>
<td>SCCB</td>
<td>subsystem connection control block MRO</td>
</tr>
<tr>
<td>SCE</td>
<td>storage element descriptor SM</td>
</tr>
<tr>
<td>SCF</td>
<td>free storage descriptor SM</td>
</tr>
<tr>
<td>SCTE</td>
<td>subsystem control table extension MRO</td>
</tr>
<tr>
<td>SDSCI</td>
<td>TD SCSCI TDP</td>
</tr>
<tr>
<td>SDTE</td>
<td>system dump table elements DU</td>
</tr>
<tr>
<td>SHRCTL</td>
<td>shared LSRPOOLs FCP</td>
</tr>
<tr>
<td>SITDLI</td>
<td>SIT DL/I extension PA</td>
</tr>
<tr>
<td>SJA</td>
<td>JVM domain anchor block SJ</td>
</tr>
<tr>
<td>SJCCH</td>
<td>Shared class cache control block SJ</td>
</tr>
<tr>
<td>SJTCB</td>
<td>J8 or J9 TCB control block SJ</td>
</tr>
<tr>
<td>SJVMS</td>
<td>Master JVM, or JVMset, control block SJ</td>
</tr>
<tr>
<td>SKT</td>
<td>scatter tables TMP</td>
</tr>
<tr>
<td>SLCB</td>
<td>subsystem logon control block MRO</td>
</tr>
<tr>
<td>SMA</td>
<td>storage manager domain anchor block SM</td>
</tr>
<tr>
<td>SMX</td>
<td>transaction storage area SM</td>
</tr>
<tr>
<td>SNE</td>
<td>Service handler element PI</td>
</tr>
<tr>
<td>SNEX</td>
<td>TCTTE signon extension TCP</td>
</tr>
<tr>
<td>SQE</td>
<td>suspend queue element SM</td>
</tr>
<tr>
<td>SSA</td>
<td>static storage areas SSA</td>
</tr>
<tr>
<td>SSAL</td>
<td>static storage address list SSA</td>
</tr>
<tr>
<td>STANCHOR</td>
<td>statistics domain anchor block ST</td>
</tr>
<tr>
<td>STATSBUF</td>
<td>log manager statistics LG</td>
</tr>
<tr>
<td>STE</td>
<td>TS sysid table entry TS</td>
</tr>
<tr>
<td>STSAFPB</td>
<td>statistics authorization facility parameter block ST</td>
</tr>
<tr>
<td>STSMF</td>
<td>statistics SMF record ST</td>
</tr>
<tr>
<td>STSTATS</td>
<td>statistics domain statistics record ST</td>
</tr>
<tr>
<td>SUA</td>
<td>subspace area SM</td>
</tr>
<tr>
<td>SUDB</td>
<td>subsystem user definition block MRO</td>
</tr>
<tr>
<td>SUSPAREA</td>
<td>unformatted SUSPEND AREAS/TOKENS DS</td>
</tr>
<tr>
<td>SYSLIB</td>
<td>DFHEJDUB EJ</td>
</tr>
<tr>
<td>SYS_TCA</td>
<td>TCA - system DLI AP</td>
</tr>
<tr>
<td>SZSDS</td>
<td>FEPI static area SZ</td>
</tr>
<tr>
<td>TACLE</td>
<td>terminal abnormal condition line entry TCP</td>
</tr>
<tr>
<td>TAH</td>
<td>task table header KE</td>
</tr>
<tr>
<td>TAS</td>
<td>task table entry - TASENTRY KE</td>
</tr>
<tr>
<td>TASK</td>
<td>unformatted DTAs DS</td>
</tr>
<tr>
<td>TCA</td>
<td>task control area DLI AP</td>
</tr>
<tr>
<td>TCH</td>
<td>KTCB table header KE</td>
</tr>
<tr>
<td>TCL</td>
<td>XM domain tclass XM</td>
</tr>
<tr>
<td>TCTENIB</td>
<td>NIB descriptor TCP</td>
</tr>
<tr>
<td>TCTESBA</td>
<td>LU6.2 send/receive buffer TCP</td>
</tr>
<tr>
<td>TCTFX</td>
<td>TCT prefix TCP</td>
</tr>
<tr>
<td>TCTLE</td>
<td>TCT line entries TCP</td>
</tr>
<tr>
<td>TCTME</td>
<td>TCT mode entry TCP</td>
</tr>
<tr>
<td>TCTSE</td>
<td>TCT system entries TCP</td>
</tr>
<tr>
<td>TCTSK</td>
<td>TCT skeleton entries TCP</td>
</tr>
<tr>
<td>TCTTE</td>
<td>TCT terminal entries TCP</td>
</tr>
<tr>
<td>TCTTECCE</td>
<td>console control element TCP</td>
</tr>
<tr>
<td>TCTTELUC</td>
<td>TCTTE LU6.2 extension TCP</td>
</tr>
<tr>
<td>TCTTETTE</td>
<td>TCTTE extension TCP</td>
</tr>
<tr>
<td>CICS Control Block</td>
<td>VERBEXIT Keyword</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>TCTTEUA</td>
<td>TCTTE user area</td>
</tr>
<tr>
<td>TD</td>
<td>user transient data</td>
</tr>
<tr>
<td>TDCUB</td>
<td>TD CI update block</td>
</tr>
<tr>
<td>TDQUB</td>
<td>TD queue update block</td>
</tr>
<tr>
<td>TDST</td>
<td>TD static storage</td>
</tr>
<tr>
<td>TDTE</td>
<td>transaction dump table elements</td>
</tr>
<tr>
<td>TDUA</td>
<td>TD UOW anchor block</td>
</tr>
<tr>
<td>TIA</td>
<td>timer domain anchor block</td>
</tr>
<tr>
<td>TIE</td>
<td>task interface element</td>
</tr>
<tr>
<td>TIOA</td>
<td>terminal I/O area</td>
</tr>
<tr>
<td>TM_LOCKS</td>
<td>read lock blocks</td>
</tr>
<tr>
<td>TMSTATIC</td>
<td>table manager static storage</td>
</tr>
<tr>
<td>TRA</td>
<td>trace domain anchor block</td>
</tr>
<tr>
<td>TRDCB</td>
<td>auxiliary trace data set DCB</td>
</tr>
<tr>
<td>TRDECB</td>
<td>auxiliary trace data set DECB</td>
</tr>
<tr>
<td>TRE</td>
<td>timer request elements</td>
</tr>
<tr>
<td>TS</td>
<td>user temporary storage</td>
</tr>
<tr>
<td>TSA</td>
<td>TS anchor block</td>
</tr>
<tr>
<td>TSBUFFER</td>
<td>TS I/O buffer</td>
</tr>
<tr>
<td>TSHANCH</td>
<td>TS shared class anchor</td>
</tr>
<tr>
<td>TSI</td>
<td>TS item descriptor</td>
</tr>
<tr>
<td>TSM</td>
<td>TS main item header</td>
</tr>
<tr>
<td>TSMNANCH</td>
<td>TS main class anchor</td>
</tr>
<tr>
<td>TSMODEL</td>
<td>TS model class anchor</td>
</tr>
<tr>
<td>TSNANCH</td>
<td>TS name class anchor</td>
</tr>
<tr>
<td>TSOANCH</td>
<td>TS ownership lock class anchor</td>
</tr>
<tr>
<td>TSQ</td>
<td>TS queue control block</td>
</tr>
<tr>
<td>TSQANCH</td>
<td>TS queue class anchor</td>
</tr>
<tr>
<td>TSS</td>
<td>TS aux section descriptor</td>
</tr>
<tr>
<td>TST</td>
<td>TS table header</td>
</tr>
<tr>
<td>TSTTE</td>
<td>TS table entry</td>
</tr>
<tr>
<td>TSW</td>
<td>TS wait element</td>
</tr>
<tr>
<td>TSX</td>
<td>TS aux item descriptor</td>
</tr>
<tr>
<td>TXN</td>
<td>XM domain transaction</td>
</tr>
<tr>
<td>TXN_DATA</td>
<td>transaction data</td>
</tr>
<tr>
<td>UCA</td>
<td>use count array</td>
</tr>
<tr>
<td>UET</td>
<td>user exit table</td>
</tr>
<tr>
<td>USER24</td>
<td>task storage - below 16 MB, user key</td>
</tr>
<tr>
<td>USER31</td>
<td>task storage - above 16 MB, user key</td>
</tr>
<tr>
<td>VCA</td>
<td>TS VSWA control area</td>
</tr>
<tr>
<td>VEMA</td>
<td>TD VSAM error message area</td>
</tr>
<tr>
<td>VSWA</td>
<td>VSAM work area</td>
</tr>
<tr>
<td>VSWA</td>
<td>TS VSAM work area</td>
</tr>
<tr>
<td>WAITLST</td>
<td>Wait list</td>
</tr>
<tr>
<td>WBABC</td>
<td>Web anchor block</td>
</tr>
<tr>
<td>WBSTC</td>
<td>Web state manager block</td>
</tr>
<tr>
<td>WCB</td>
<td>Web service control block</td>
</tr>
<tr>
<td>WHB</td>
<td>Web service header block</td>
</tr>
<tr>
<td>WQP</td>
<td>Domain wait queue</td>
</tr>
<tr>
<td>WRB</td>
<td>Web service resource block</td>
</tr>
</tbody>
</table>

Appendix A. SDUMP contents and IPCS CICS VERBEXIT keywords
<table>
<thead>
<tr>
<th>CICS Control Block</th>
<th>VERBEXIT Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMA</td>
<td>XM</td>
</tr>
<tr>
<td>XM_XB</td>
<td>XM</td>
</tr>
<tr>
<td>XRH</td>
<td>TS</td>
</tr>
<tr>
<td>XRP_ACTS</td>
<td>XRF</td>
</tr>
<tr>
<td>XRP_ALTS</td>
<td>XRF</td>
</tr>
<tr>
<td>XRP_HLTH</td>
<td>XRF</td>
</tr>
<tr>
<td>XRP_XRSA</td>
<td>XRF</td>
</tr>
<tr>
<td>XRPSTAT</td>
<td>XRF</td>
</tr>
<tr>
<td>ZEPD</td>
<td>TCP</td>
</tr>
</tbody>
</table>

XM domain anchor block
XM domain browse element
TS aux record header
XRP active status area
XRP alternate status area
XRP health area
XRP anchor area
XRP static storage
TC module entry list
Appendix B. Summary data for PG and US keywords

This section 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 Appendix A, "SDUMP contents and IPCS CICS VERBEXIT keywords," on page 321 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)

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG Domain Status</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• Initializing</td>
</tr>
<tr>
<td></td>
<td>• Initialized</td>
</tr>
<tr>
<td></td>
<td>• Quiescing</td>
</tr>
<tr>
<td></td>
<td>• Quiesced</td>
</tr>
<tr>
<td></td>
<td>• Terminating</td>
</tr>
<tr>
<td></td>
<td>• Terminated</td>
</tr>
<tr>
<td>Autoinstall status</td>
<td>Either active or inactive.</td>
</tr>
<tr>
<td>Autoinstall catlg status</td>
<td>Autoinstall catalog status, one of the following:</td>
</tr>
<tr>
<td></td>
<td>• All</td>
</tr>
<tr>
<td></td>
<td>• Modify</td>
</tr>
<tr>
<td></td>
<td>• None.</td>
</tr>
<tr>
<td>Autoinstall exit name</td>
<td>Autoinstall exit name.</td>
</tr>
<tr>
<td>Attempted autoinstalls</td>
<td>Number of attempted autoinstalls in decimal.</td>
</tr>
<tr>
<td>Failed autoinstalls</td>
<td>Number of failed autoinstalls in decimal.</td>
</tr>
<tr>
<td>Rejected autoinstalls</td>
<td>Number of rejected autoinstalls in decimal.</td>
</tr>
<tr>
<td>XRSINDI active</td>
<td>Status of user exit, either Y or N.</td>
</tr>
<tr>
<td>Exec calls allowed</td>
<td>Either Y or N.</td>
</tr>
<tr>
<td>System LLE chain head</td>
<td>Address of system LLE chain head, zero if no chain exists.</td>
</tr>
<tr>
<td>PGWE chain head</td>
<td>Address of PGWE chain head, or zero if no chain exists.</td>
</tr>
<tr>
<td>Stats last - 1st word</td>
<td>Statistics last reset time using GMT (on two lines).</td>
</tr>
<tr>
<td>Reset time - 2nd word</td>
<td>Second part of last reset time.</td>
</tr>
<tr>
<td>SM access token</td>
<td>SM access token value.</td>
</tr>
<tr>
<td>SM isolation token</td>
<td>SM isolation token value.</td>
</tr>
<tr>
<td>Storage protect</td>
<td>Either Y or N.</td>
</tr>
<tr>
<td>Cold start</td>
<td>Either Y or N.</td>
</tr>
<tr>
<td>Recovery complete</td>
<td>Either Y or N.</td>
</tr>
</tbody>
</table>

System LLE Summary

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLE-ADDR</td>
<td>LLE address.</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Program name.</td>
</tr>
<tr>
<td>PPTE-ADD</td>
<td>PPTE address.</td>
</tr>
</tbody>
</table>
### 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 (or Ada)
  - **C**: C370
  - **COB**: Cobol
  - **CO2**: Cobol 2
  - **LE3**: Le370
  - **PLI**: PL/I.
- **LANG DED**: Language deduced, one of the following:
  - **NDD**: Not deduced
  - **ASS**: Assembler (or Ada)
  - **C**: C370
  - **COB**: Cobol
  - **CO2**: Cobol 2
  - **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 D (disabled).
- **DATA LOC**: Data location, either A (any location) or B (below 16M).
- **EXEC KEY**: Execution key, either C (CICS) or U (user).
- **DPL SUBS**: DPL subset, either DP (DPL subset) or F (full API).
- **RE LOAD**: Indicator of whether this is a reload program, either Y or N.
- **LOAD STAT**: Load status, one of the following:
  - **L**: Loaded
  - **NL**: Not loadable
  - **ND**: Not loaded.
- **HOLD STAT**: CICS hold status, either C (loaded for CICS lifetime) or T (task lifetime).
- **USE COUNT**: Use count in decimal, blank if 0.
- **LOCK OWNER**: Transaction number of locking program.
- **PGWE CHAIN**: Indicator of presence of any PGWEs, either Y or N.
### 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

---

Appendix B. Summary data for PG and US keywords 341
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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAN NUM</td>
<td>Transaction number.</td>
</tr>
<tr>
<td>PRINCIPAL TOKEN</td>
<td>Principal token, if any.</td>
</tr>
<tr>
<td>SESSION TOKEN</td>
<td>Session token, if any.</td>
</tr>
<tr>
<td>EDF TOKEN</td>
<td>EDF token, if any.</td>
</tr>
</tbody>
</table>

USUD summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOKEN</td>
<td>User token.</td>
</tr>
<tr>
<td>USERID</td>
<td>User identifier.</td>
</tr>
<tr>
<td>GROUPID</td>
<td>Group identifier.</td>
</tr>
<tr>
<td>ADDCOUNT</td>
<td>Adduser use count.</td>
</tr>
<tr>
<td>TRNCOUNT</td>
<td>Transaction use count.</td>
</tr>
<tr>
<td>OPID</td>
<td>Operator identifier.</td>
</tr>
<tr>
<td>CLASSES</td>
<td>A bitmap expressing the operator classes in order 24 to 1.</td>
</tr>
<tr>
<td>PRTY</td>
<td>Operator priority.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Timeout interval in hours and minutes (hh:mm).</td>
</tr>
<tr>
<td>ACEE</td>
<td>Address of ACEE.</td>
</tr>
<tr>
<td>XRFSOFF</td>
<td>XRF user signon. Can be NOFORCE or FORCE.</td>
</tr>
<tr>
<td>USERNAME</td>
<td>User name.</td>
</tr>
</tbody>
</table>
The CICS Transaction Server for z/OS library

The published information for CICS Transaction Server for z/OS is delivered in the following forms:

The CICS Transaction Server for z/OS Information Center
The CICS Transaction Server for z/OS Information Center is the primary source of user information for CICS Transaction Server. The Information Center contains:

- Information for CICS Transaction Server in HTML format.
- Licensed and unlicensed CICS Transaction Server books provided as Adobe Portable Document Format (PDF) files. You can use these files to print hardcopy of the books. For more information, see “PDF-only books.”
- Information for related products in HTML format and PDF files.

One copy of the CICS Information Center, on a CD-ROM, is provided automatically with the product. Further copies can be ordered, at no additional charge, by specifying the Information Center feature number, 7014.

Licensed documentation is available only to licensees of the product. A version of the Information Center that contains only unlicensed information is available through the publications ordering system, order number SK3T-6945.

Entitlement hardcopy books
The following essential publications, in hardcopy form, are provided automatically with the product. For more information, see “The entitlement set.”

The entitlement set
The entitlement set comprises the following hardcopy books, which are provided automatically when you order CICS Transaction Server for z/OS, Version 3 Release 1:

- Memo to Licensees, GI10-2559
- CICS Transaction Server for z/OS Program Directory, GI10-2586
- CICS Transaction Server for z/OS Installation Guide, GC34-6426
- CICS Transaction Server for z/OS Licensed Program Specification, GC34-6608

You can order further copies of the following books in the entitlement set, using the order number quoted above:

- CICS Transaction Server for z/OS Release Guide
- CICS Transaction Server for z/OS Installation Guide
- CICS Transaction Server for z/OS Licensed Program Specification

PDF-only books
The following books are available in the CICS Information Center as Adobe Portable Document Format (PDF) files:

CICS books for CICS Transaction Server for z/OS
General
CICS Transaction Server for z/OS Migration from CICS TS Version 1.3, GC34-6423
CICS Transaction Server for z/OS Migration from CICS TS Version 2.2, GC34-6424
CICS Transaction Server for z/OS Installation Guide, GC34-6426

Administration
CICS System Definition Guide, SC34-6428
CICS Customization Guide, SC34-6429
CICS Resource Definition Guide, SC34-6430
CICS Operations and Utilities Guide, SC34-6431
CICS Supplied Transactions, SC34-6432

Programming
CICS Application Programming Guide, SC34-6433
CICS Application Programming Reference, SC34-6434
CICS System Programming Reference, SC34-6435
CICS Front End Programming Interface User's Guide, SC34-6436
CICS C++ OO Class Libraries, SC34-6437
CICS Distributed Transaction Programming Guide, SC34-6438
CICS Business Transaction Services, SC34-6439
Java Applications in CICS, SC34-6440
JCICS Class Reference, SC34-6001

Diagnosis
CICS Problem Determination Guide, SC34-6441
CICS Messages and Codes, GC34-6442
CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Trace Entries, SC34-6443
CICS Supplementary Data Areas, GC34-6905

Communication
CICS Intercommunication Guide, SC34-6448
CICS External Interfaces Guide, SC34-6449
CICS Internet Guide, SC34-6450

Special topics
CICS Recovery and Restart Guide, SC34-6451
CICS Performance Guide, SC34-6452
CICS IMS Database Control Guide, SC34-6453
CICS RACF Security Guide, SC34-6454
CICS Shared Data Tables Guide, SC34-6455
CICS DB2 Guide, SC34-6457
CICS Debugging Tools Interfaces Reference, GC34-6908

CICSPlex SM books for CICS Transaction Server for z/OS
General
CICSPlex SM Concepts and Planning, SC34-6459
CICSPlex SM User Interface Guide, SC34-6460
CICSPlex SM Web User Interface Guide, SC34-6461

Administration and Management
CICSPlex SM Administration, SC34-6462
CICSPlex SM Operations Views Reference, SC34-6463
CICSPlex SM Monitor Views Reference, SC34-6464
CICSPlex SM Managing Workloads, SC34-6465
CICSPlex SM Managing Resource Usage, SC34-6466
CICSPlex SM Managing Business Applications, SC34-6467

Programming
CICSPlex SM Application Programming Guide, SC34-6468
CICSPlex SM Application Programming Reference, SC34-6469
Diagnosis

CICSPlex SM Resource Tables Reference, SC34-6470
CICSPlex SM Messages and Codes, GC34-6471
CICSPlex SM Problem Determination, GC34-6472

CICS family books

Communication

CICS Family: Interproduct Communication, SC34-6473
CICS Family: Communicating from CICS on System/390, SC34-6474

Licensed publications

The following licensed publications are not included in the unlicensed version of the Information Center:

CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Supplementary Data Areas, GC34-6905
CICS Debugging Tools Interfaces Reference, GC34-6908

Other CICS books

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 3 Release 1.

Designing and Programming CICS Applications, SR23-9692
CICS Application Migration Aid Guide, SC33-0768
CICS Family: API Structure, SC33-1007
CICS Family: Client/Server Programming, SC33-1435
CICS Transaction Gateway for z/OS Administration, SC34-5528
CICS Family: General Information, GC33-0155
CICS 4.1 Sample Applications Guide, SC33-1173
CICS/ESA 3.3 XRF Guide, SC33-0661

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager® softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the Transaction Processing and Data Collection Kit CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Updates to the softcopy are clearly marked by revision codes (usually a # character) to the left of the changes.
Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.
Index

A
abbreviated-format trace 256
abend codes
  transaction 25
    AICA 26
    ASRA 26
    ASRB 27
    CICS 26
  destination 25
  documentation 26
  interpretation 26
  product other than CICS 26
  user 26
ABEND symptom keyword 7
abends
  AICA 144
  DBCTL interface 35
dump not made when expected 173
exception trace entry 38
investigating 37
  the documentation you need 37
looking at the symptom string 38
symptom keyword 7
transaction 25
  AICA 26
  ASRA 26
  ASRB 27
  worksheet 35
access method
  BSAM 60
determining the type in use 59
intersystem communication 59
ISMM 59
possible reason for stalled system 109
VTAM
terminal control waits 133
terminal waits 60
addressing exception 30
AEYD
causes 27
AICA abend
  probable cause 26
  PSW 297
  registers 297
AICA abends 144
AID chain
  investigating tasks that have not started 188
  locating in the formatted system dump 188
AIDTRMID (symbolic ID of terminal) 189
AIDTRNID (transaction ID) 189
AITM resource name 114
ALLOCATE resource type 113, 134
alternate system waits 139
ALTPAGE attribute 178, 182
ALTSCREEN attribute, 178, 182
Any_MBCB resource type 113
Any_MRCB resource type 113
AP_INIT resource type 113
AP_QUIES resource type 113
AP_TERM resource type 113
APAR (authorized program analysis report)
  authorization 315
closing 317
documentation needed 315
process 315
submitting 316
APPC (LUTYPE6.2), range of devices 58
application programs
dynamic storage 302
storage areas 302
arithmetic exceptions 30
  investigating 30
ASIS option 179
ASRA abend
  causes 26
  execution key 297
  PSW 297
  registers 297
ASRB abend
  causes 27
  execution key 297
  PSW 297
  registers 297
ASRD abend
  causes 27
  PSW 297
  registers 297
assembler programs
  locating the DFHEISTG storage 302
assemblers
ersors in output 6
asynchronous processing 66
ASYNRESP resource name 114
ATCHMSUB, resource name 115
ATI (automatic transaction initiation) 64
autoinitiated tasks
  excessive numbers shown in statistics 14
  automatic initiate descriptor (AID)
    identifying the related task 189
    identifying the terminal 189
    investigating tasks that have not started 188
automatic transaction initiation (ATI)
task produced no output 185, 188
  looking at the AID chain 188
  looking at the ICE chain 188
  resource not available 188
  task not yet scheduled to start 188
  automatic transaction initiation session status 64
auxiliary trace 255
abbreviated-format 256
advantages 246
characteristics 245
controlling 247
data sets 245
destination 245
CICS stalled (continued)
during a run 107, 144
during initialization 106
during quiesce 109
during termination 109
effect of ICV parameter 107
effect of ICVR parameter 108
effect of ICVTSD parameter 108
effect of MXT parameter 108
exclusive control of volume conflict 109
investigating the reason for the stall 8
messages 8
on cold start 106
on emergency restart 106
on initial start 106
on warm start 106
PLT initialization programs 107
PLT shutdown programs 110
possible causes 8
specific regions 13
system definition parameters wrong 107
CICS system abends
action of global trap/trace exit 306
CICS system dump following 264
CSMT log messages 8
exception trace entry 38
information needed by the Support Center 37
investigating 37
looking at the symptom string 38
messages 8
the documentation you need 37
CICS system dumps
data not formatted correctly 176
destination 263
dispatcher domain storage areas 93
dump not made on CICS system abend 173
following CICS system abend 264
following transaction abend 264
formatting 285
selectivity 285
formatting keywords 287
formatting levels 287
from global trap/trace exit 306
global suppression 174, 261
in problem determination 261
interactive problem control system (IPCS) 285
internal trace table 37
investigating CICS system abends 37
investigating waits 51, 53
kernel domain storage areas
CICS system abends 37
error code 42
error data 43
error type 42
failing program 42
information provided 40
kernel error number 42
point of failure 42
PSW at time of error 43
registers at time of error 43
task error information 41
CICS system dumps (continued)
kernel domain storage areas (continued)
task summary 40
tasks in error 40
waits for resource locks 93
locating the AID chain 188
locating the ICE chain 188
lock manager domain storage areas 93
looking at the symptom string 38
of remote CICS regions 66
precautions using dump formatting keywords 176
statistics 276
storage manager domain storage areas 68
storage violation 195, 196, 200
suppression by user exit program 174
suppression for individual transactions 175, 261
suppression for specific dump codes 261
system dump code option 275
system dump codes 261
temporary storage control blocks 69
terminal control storage areas 59
CICS, resource name 118
CICS610 dump exit
DEF parameter 287
JOB parameter 286
keyword parameter 286
classification of problems 7
COBOL programs
task global table 302
working storage 302
common system area (CSA)
in transaction dump 298
locating the AID chain 188
optional features list 299
compilers
errors in output 6
component tracing
identifying codes 235
precautions when selecting 171
setting special trace levels 235
setting standard trace levels 235
control interval (CI)
exclusive control deadlock 89
exclusive control waits 88
CPI resource name 115
CRTE and uppercase translation 179
CSAOPFL 299
CSASSI2 resource name 113
CSATODTU 76
CSFE DEBUG transaction
global trap/trace exit 305
storage checking 197
TRAP operand 305
CSFE transaction
checking the programming logic 191
storage freeze option 191
CSFE ZCQTRACE transaction
dumps of builder parameter set 283
CSMT log
abend messages 3, 8, 10
terminal error messages 10, 58
data corruption
  - bad programming logic 190
  - incorrect mapping to program 190
  - incorrect mapping to terminal 191
    - attributes of fields 191, 192
    - DARK field attribute 191
    - modified data tag 191
    - symbolic map 191
  - incorrect records in file 190
  - missing records in file 190
  - possible causes 189
data exception 29
DATABUFFERS parameter of FILE resource
  definition 83
DB2 migration considerations
  DSNTIAR 33
DB2 resource type 114
  DB2_INIT resource type 114
  DB2CDISC resource type 114
  DB2EDISA resource type 114
  DB2START, resource name 122
DBCTRL (database control)
  abends 35
    - connection fails 128
    - disconnection fails 128
    - immediate disconnection 129
    - orderly disconnection 129
    - waits 128
  DBCTRL resource type 114, 128
  DBDXEOT resource type 114
  DBDXINT resource type 114
  DEBUGUSER resource name 115
  DCT resource name 121
deadlock time-out interval
  EXEC CICS WRITEQ TS command 69
    - interval control waits 75
    - task storage waits 67
deadlocks
  resolving 101
  resolving in a sysplex 105
debugging
  IRC problems 205
  multiregion operation problems 205
DEF parameter of CICS640 dump exit 287
destination control table (DCT)
  - extrapartition transient data destination 136
  - logically recoverable queues 136
  DFHAILN resource type 114
  DFHAUX 245
  DFHBUX 245
  DFHCPI resource type 115
  DFHDMPA dump data set 262
  DFHDMPB dump data set 262
  DFHUD640 job log 297
DSA (dynamic storage area)
current free space  68
storage fragmentation  69
DSNTIAR  33
DSSR gate of dispatcher domain
tracing the functions  52
interpreting the trace table  52
tracing the input and output parameters  52
DTCHMSUB, resource name  115
dump codes
checking the attributes  175
DUMPSCOPE option  264, 265
RELATED attribute  264, 265
storage violation  195
suppression of system dumps  261
suppression of transaction dumps  261
system  175, 261
CICS termination option  275
maximum dumps option  275
NOSYS_DUMP attribute  175
options  275
RELATED dumping option  275
SYSDUMP attribute  175
system dumping option  275
transaction  175, 261
CICS termination option  275
format  263
maximum dumps option  275
NOTRANDUMP attribute  175
options  275
RELATED dumping option  275
system dumping option  275
TRAN_DUMP attribute  175
transaction dumping option  275
dump data sets
attributes  262
AUTOSWITCH status  262
CLOSED status  262
current status  262
DFHDMPA  262
DFHDMPB  262
inquiring on  262
NOAUTOSWITCH status  262
OPEN status  262
setting  262
switch status  262
dump domain
XDUREQ global user exit  174
dump table
examples  278, 280
options  175, 275
loss of additions and changes  275
preservation of additions and changes  275
statistics  276
current count  276
reset  276
system dumps suppressed  276
system dumps taken  276
times dump code action taken  276
transaction dumps suppressed  276
transaction dumps taken  276
dump table (continued)
suppression of dumping  175
system  280
temporary entries  275, 277
transaction  278
DUMP, system initialization parameter  174, 261
DUMPDS, system initialization parameter  262
dumping in a sysplex  265
dumps
CFDT list structure dump  281
controlling
CEMT transaction  263
dump codes  261
dump tables  261
examples  278, 280
EXEC CICS commands  263
selective dumping of storage  264, 276
specifying dump options  275
using an undefined dump code  277
controlling dump action  261
current dump ID  280
dump output is incorrect
data not formatted correctly  176
dump not made on abend  173
investigating  173
some dump IDs missing from the sequence of
dumps  175
wrong CICS region  173
dumps  18
events that can cause dumps  263
formatting a CFDT pool dump  303
formatting a named counter pool dump  304
formatting a shared temporary storage pool
dump  304
formatting keywords  287
formatting levels  287
IDs missing from the sequence of dumps  175
in a sysplex  265
in problem determination  261
looking at the symptom string  38
named counter list structure dump  281
options  275
requesting dumps  263
setting the dumping environment  261
shared temporary storage list structure dump  282
suppressing  174, 262
DUMPSCOPE dump code option  264, 265
DUMPDSW, system initialization parameter  262
DURETRY, system initialization parameter  263

E
EARLYPLT resource name  115
ECB (event control block)
EXEC CICS POST command  75
finding the address  131
invalid address, task control waits  131
posting after task is canceled  199
PSTDECB  96
storage violations  199
valid address, task control waits  132
EXEC CICS WRITEQ TS command 69
   NOSUSPEND 69
   REWRITE option 69
EXEC interface block (EIB)
   EIBFN 298
EXECADDR resource name 115
EXECSTRN resource name 115
execution diagnostic facility (EDF)
   investigating loops 159
   use in investigating no task output 186
   waits 129
EDF resource type 115
EDSA (extended dynamic storage area)
   current free space 68
   storage fragmentation 69
   in last command identification 301
ENQUEUE resource type 115
ENQUEUE on single server resource 132
FC environment rebuild 85
F
FCACWAIT resource type 116
FCBFWAIT resource type 83, 116
FCCAWAIT resource type 83, 116
FCCFQR resource type 84, 116
FCCFQS resource type 84, 116
FCCRWAIT resource type 117
FCFLRECD resource name 116
FCFLUMTL resource name 116
FCFSWAIT resource type 84, 117
FCINWAIT resource type 117
FCIOWAIT resource type 85, 117
FCIRWAIT resource type 85, 117
FCPSWAIT resource type 85, 117
FCQUIES resource type 86, 117
FCRAWAIT resource type 86, 117
FCRBWAIT resource type 86, 117
FCRDWAIT resource type 88, 117
FCRPWAIT resource type 87, 117
FCRRWAIT resource type 87, 117
FCRVWAIT resource type 87, 117
FCSRSUSP resource type 85, 117
FCXWAIT resource type 85, 117
FCXWAIT resource type 88, 118
file accesses, excessive 14
file control waits 80
   BDAM record locking 92
   drain of RLS control ACB 86
   ESDS write lock 92
   exclusive control conflict 88
   exclusive control deadlock 89
   FC environment rebuild 85

EXCLOGER resource name 114
exclusive control of volume conflict 109
EXEC CICS ABEND command 264
EXEC CICS DELAY command 75
EXEC CICS DUMP TRANSACTION command 263, 264, 267
EXEC CICS ENTER TRACENUM command 228
EXEC CICS INQUIRE TASK
   SUSPENDTYPE field 51
   SUSPENDVALUE field 51
EXEC CICS PERFORM DUMP command 263
EXEC CICS POST 75
EXEC CICS READ UPDATE command 89
EXEC CICS RETRIEVE WAIT command 75
EXEC CICS REWRITE command 89
EXEC CICS START command 75, 162
EXEC CICS STARTBR command 89
EXEC CICS WAIT EVENT command 75
EXEC CICS WRITE command 90
EXEC CICS WRITE MASSINSERT command 90
F
file control waits (continued)
  file state changes 84
  KSDS range lock 92
  process non-recoverable requests 86
  process recoverable requests 86
  RLS control ACB access 83
  VSAM buffer unavailable 83
  VSAM completing update processing 84
  VSAM I/O 85, 87
  VSAM I/O waits (RLS) 87
  VSAM load mode lock 92
  VSAM record locking by CICS 91
  VSAM string unavailable 85
  VSAM transaction IDs 88
  VSAM upgrade set activity 84, 86
  wait for dynamic RLS restart 87
  wait for FC initialization 87
files
  no task output 187
  first failure data capture 228, 245
  FOREVER resource type 118
  formatting CICS system dumps 285
  keywords 287
front end programming interface (FEPI)
  dump control option 293
  FEPI waits 141
function shipping 66

G
  global catalog data set (GCD)
    dump table options 275
    effect of redefinition on dump table 275
  global ENQUEUE 105
  global trap/trace exit 200, 305
  actions the exit can take 306
  activating and deactivating the exit 305
  coding 307
  establishing the exit 305
  information passed to the exit 305
  program check handling 307
  replacing a trap exit 305
  uses 305
  work area 307
  GTFTR, system initialization parameter 247

H
  HETYPE field 51
  HVALUE field 51

I
  I/O buffers, transient data
    all in use 138
  IBM Support Center
    call receipt 311
    Call Receipt 312
    dealing with the Center 311
    problem reporting 311
    use of RETAIN database 7, 313
  IBM Support Center (continued)
    when to contact 311
  ICE (interval control element) 188
  ICE expiration 76
  ICETRNID transaction ID 188
  ICEXPIRY resource type 118
  ICEXTOD expiration time 188
  ICEXTOD value 76
  ICGTWAIT resource type 57, 75, 118
  ICMIDNTE resource type 118
  ICV, system initialization parameter
    possible cause of CICS stall 107
  ICVR, system initialization parameter
    non-yielding loops 144
    possible cause of CICS stall 108
    tight loops 144
  ICVTSD, system initialization parameter
    possible cause of CICS stall 108
  ICWAIT resource type 57, 75, 118
  IIOP
    waits 135
  incorrect output
    abend codes 18
    application did not work as expected 183
    BMS mapping 191
      attributes of fields 191, 192
      DARK field attribute 191
      MDT 191
      modified data tag 191
      symbolic map 191
    change log 18
    checking for bad data in a file 190
    checking the mapping from file to program 190
    checking the programming logic 190
      desk checking 190
      using CEBR 191
      using CECI 190
      using CEDF 190
      using interactive tools 190
    databases 21
    error messages 18
    files 21
    incorrect output read from VSAM data set 182
    investigating 169
    link-edit maps 17
    monitoring 19
    no output obtained 184
      ATI tasks 185, 188
      disabling the transaction 187
      explanatory messages 184
      finding if the task ran 185
      looking at files 187
      looking at temporary storage queues 187
      looking at transient data queues 187
      possible causes 184
      START PRINTER bit on write control
        character 185
      task not in system 185
      task still in system 185
      testing the terminal status 184
      using CEI 187
incorrect output (continued)
no output obtained (continued)
using execution diagnostic facility 186
using statistics 186, 187
using trace 185
passed information 20
printed output wrong 176
source listings 17
statistics 19
symptom keyword 7
symptoms 11
temporary storage 20
terminal data 20
terminal output wrong 176
trace 21
trace data wrong 170
trace destination wrong 169
trace entries missing 171
trace output wrong 169
transaction inputs and outputs 19
transient data 20
unexpected messages 11
user documentation 17
wrong CICS components being traced 171
wrong output obtained 189
possible causes 189
wrong tasks being traced 171
INCORROUT symptom keyword 7
INDEXBUFFERS parameter of FILE resource definition 83
information sources 17
INFORMATION/ACCESS licensed program 7
INITIAL resource name 113
initialization stall 106
INQUIRE UOWENQ command deadlock diagnosis 105
INQUIRE_ resource name 123
interactive problem control system (IPCS) analyzing CICS system dumps 285
CICS system abends 37
CICS640 dump exit 286
internal trace 255
abbreviated-format 256
characteristics 245
controlling 247
destination 245
exception trace destination 245
extended-format 252
formatting 251
interpreting 252, 256
short-format 255
status 245, 247
trace entries missing 172
trace table size 245
changing the size dynamically 245
wrapping 245
intersystem communication (ISC)
poor performance 162
waits 66, 134
interval control
element 76
interval control (continued)
performance considerations 162
waits 75
deadlock time-out interval 75
systematic investigation 76
using trace 76
interval control element (ICE) 188
INTTR, system initialization parameter 247
IRC (interregion communication) 205
poor performance 162
waits 66, 134
IRLINK resource type 57, 66, 118
ISMM access method 59
J
JOB parameter of CICS640 dump exit 286
JOURNALs resource name 116
JVM tracing 238
defining 238
interpreting 243
selecting trace options 239
JVM_POOL resource name 115
JVM_POOL wait 125
K
Katakana terminals 181
mixed English and Katakana characters 181
KCADDR resource name 116
KCCOMPAT resource type 118, 119, 130, 131
resource names 131
CICS 131
LIST 131
SINGLE 131
SUSPEND 131
TERMINAL 57, 131
KCSTRING resource name 116
kernel domain
information given in dump 40
error code 42
error data 43
error table 41
error type 42
failing program 42, 45
kernel error number 42
point of failure 42
PSW at time of error 43
registers at time of error 43
storage addressed by PSW 44
storage addressed by registers 44
task error information 41
task summary 40
tasks in error 40, 41
linkage stack 45
identifying the task in error 45
stack entries 298
storage areas 40
keyword parameter of CICS640 dump exit 286
last command identification 300
last statement identification 301
LATE_PLT resource name 115
LATE_PLT resource type 119
level-1 trace points 227
level-2 trace points 227
level-3 trace points 227
LG_DEFER resource type 119
LG_FORCE resource type 119
LG_MGRST resource name 119
LGDELALL resource type 119
LGDELRAN resource type 119
LGENDBLK resource type 119
LGENDCRS resource type 119
LGHARTBT resource type 119
LGREDBLK resource type 119
LGREDCRS resource type 119
LGSTRBLK resource type 119
LGSTRCRS resource type 119
LGWRITE resource type 119
link editor
   errors in output 6
LIST resource name 118, 123
LMQUEUE resource name 93, 113
loader domain (LD)
   program storage map 46
   waits 93
lock manager domain (LM)
   involvement in waits 93
   identifying the lock owning task 93
   investigating 93
log manager problems
   categories of 207
   corrupt system log 221
   diagnostic run, of CICS 221
   exceeding the capacity of a log stream 208
   problems in the MVS logger 213
logon rejection 178
LOGSTRMS resource name 116
LOOP symptom keyword 7
loops
   CICS detected 143
   debugging with interactive tools 159
   identifying the point of entry 147
   in CICS code 143
   in system code 156
   investigating 143
      looking at the evidence 147
      techniques 146, 157
   investigating by modifying your program 159
   looking at the transaction dump 156
   non-yielding 143
      characteristics 144
      finding the cause 157
      investigating 146
      possible causes 157
   potential consumption of storage 68
   potential consumption of temporary storage 70
   symptom keyword 7
loops (continued)
symptoms 12, 13, 143
   CICS region stalled 13
   CPU usage high 13
      reduced activity at terminals 13
      repetitive output 14
      short on storage 14
      system busy symbol 13
tight 143
   characteristics 144
   finding the cause 157
   identifying an instruction in the loop 156
   investigating 146
   possible causes 157
types 143
   using CEBR 159
   using CECI 159
   using CEDF 159
   using the CEMT transaction 14
   using trace 146, 157
yielding 143
   characteristics 145
   finding the cause 158
   investigating 157
   possible causes 158
   useful documentation 158
LOT_ECB resource name 114
lowercase characters 178
LUTYPE6.1, range of devices 58

M
MAXSOCKETS resource name 120
MBCB_xxx resource type 119
MESSAGE symptom keyword 7
messages
   absence of, when expected 13
   CICS under stress 14
destination
      CDBC 10
      CSMT 8, 10
      CSTL 10
      DFHAC2008 187
      DFHSM0131 15, 165
      DFHSM0133 15, 165
      DFHSR0601 8
      DFHST0001 8
dump formatting error 194
   preliminary checks 3
   short on storage 165
   sources 4
   storage violation 194, 196
   symptom keyword 7
terminal errors 58
   transaction abend 25
   transaction disabled 187
   unexpected 11, 178
MISCELANEOUS resource name 120
missing trace entries 171
module index
   in transaction dump 300
MONITOR POINT command 19
monitoring point 19
MRCC_xxx resource type 119
MRO waits 66, 134
MROQUEUE resource name 114
MSBRETRN, resource name 115
multiregion operation using IRC 205
multiregion operation waits 66, 134
MVS ABEND macro 27
MVS console
  CICS termination message 8
MVS logger availability check 208
MVS RESERVE locking
    CEDA in a wait state 140
    CESN in a wait state 140
    CICS system stalls 109
effect on CICS performance 167
transient data extrapartition waits 136
VSAM I/O waits 85
waits during XRF takeover 140
waits on resource type XRPUTMSG 140
MXT (maximum tasks value)
effect on performance 163
kernel task summary 41
possible cause of CICS stall 108
reason for task failing to start 10
waits 96
XM_HELD resource type 96
MXT resource type 119

PAGESIZE attribute, TYPETERM 182
PC, communication with CICS 58
PERFM symptom keyword 7
performance
  bottlenecks 161, 167
    dispatch, suspend and resume cycle 162
    initial attach to the dispatcher 162
    initial attach to the transaction manager 161
    initial dispatch 162
    dispatch, suspend and resume cycle 166
    extrapartition transient data 167
    initial attach to the dispatcher 163
    initial attach to the transaction manager 162
    initial dispatch to the dispatcher 165
    interval control delays 162
MXT limit 163
performance class monitoring 165
poor
  at peak system load times 9
  finding the bottleneck 161
  investigating 161
  lightly loaded system 9
  possible causes 9
  symptom keyword 7
  symptoms 9, 12, 14, 161
remote system status 162
system loading 167
task control statistics 163
task priority 165
task time-out interval 167
terminal status 162
use of MVS RESERVE 167
using trace 164
performance class monitoring 165
PIIS resource type 119
PL/I application programs
  locating the DSA chain 302
PMR (problem management record) 312
preliminary checks
  all functions fully exercised 6
  any changes to the application 5
  any previous success 3, 5
  changes since last success 4
  hardware modification 4

output (continued)
none obtained (continued)
  START PRINTER bit on write control
    character 185
  task not in system 185
  task still in system 185
  testing the terminal status 184
  using CECI 187
  using execution diagnostic facility 186
  using statistics 186, 187
  using trace 185
repetitive 14
wrong 189
  possible causes 189

N
NetView 110
networks
  messages 58
  preliminary checks 5
NOSYSDUMP, system dump code attribute 175
NOTRANDUMP, transaction dump code attribute 175

O
ONC/RPC 7
open transaction environment
  TCB stealing 126
OPEN_DEL resource name 115
OPEN_DEL wait 126
OPENPOOL resource name 115
OPENPOOL wait 125
operation exceptions 29
output
  absence when it is expected 12
  none obtained 184
  ATI tasks 185, 188
disabling the transaction 187
explanatory messages 184
finding if the task ran 185
looking at files 187
looking at temporary storage queues 187
looking at transient data queues 187
possible causes 184
preliminary checks (continued)
changes since last success (continued)
initialization procedure  5
modified application  5
new application  5
PTF (program temporary fix)  4
resource definitions  5
common programming errors  6
failure at specific times of day  4
intermittent failures  4
interval control waits  75
messages  3
network related errors  5
many terminals  5
single terminal  5
no previous success  6
output from assembler  6
output from compiler  6
output from link editor  6
output from translator  6
reproducible problems  3
caused by poor system definition  4
related to applications  4
related to system loading  4
terminal waits  58
printers
no output  185
printed output wrong  176, 181
unexpected line feeds and form feeds  182
write control character  185
privileged operation  29
PRM resource name  115
problem classification  7
problem determination
FEPI waits  141
problem management record (PMR)  312
problem reporting
documentation needed  313
IBM Program Support  311
information needed  312
report sheet  311
problems in the MVS logger  213
processors
usage high  13
PROFILE definition
SCRNSTYPE attribute  178, 182
UCTRAN attribute  179
program check
addressing exception  30
arithmetic exceptions  30
investigating  30
cause of ASRA abends  26
data exception  29
execution exception  29
investigating  27
next sequential instruction  27
operation exception  29
outside of CICS  28
possible types  28
privileged operation  29
protected by  29
program check (continued)
specification exception  30
system interrupts  30
wild branch  28
program check and abend tracing  231
program control waits  93
program interrupt code (PIC)
addressing exception  30
arithmetic exceptions  30
data exception  29
execution exception  29
interpretation  28
operation exception  29
privileged operation  29
program list table (PLT)
programs executing during CICS quiesce  110
transient data waits  107, 135
PROGRAM resource type  119, 120
program status word (PSW)  26
programming errors
preliminary checks  6
programs
information for current transaction  300
loops  143
problems with loading  93
representation in linkage stack  45
storage  300
protection exception  29
dealing with  31
possible causes  32
PSB (program specification block)
load I/O  96
PSINQE ECB resource name  123
PSOP2ECB resource name  123
PSTDECB  96
PSUNBECB resource name  123
PSW (program status word)
at time of error  43
CICS system abends  43
description  26
finding the offset of a failing instruction  46
format  46
in transaction abend control block  299
in transaction dump  297
PTF  317
PTF level  46
Q
QSAM  136, 167
QUIESCE resource name  115
quiesce stall  109
R
RCP_INIT resource type  120
RDSA resource type  120
<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>120</td>
</tr>
<tr>
<td>record locking</td>
<td></td>
</tr>
<tr>
<td>BDAM data sets</td>
<td>92</td>
</tr>
<tr>
<td>VSAM data sets</td>
<td>91</td>
</tr>
<tr>
<td>registers</td>
<td></td>
</tr>
<tr>
<td>at time of error</td>
<td>43</td>
</tr>
<tr>
<td>CICS system abends</td>
<td>43</td>
</tr>
<tr>
<td>data addressed at the time of error</td>
<td>44</td>
</tr>
<tr>
<td>in transaction abend control block</td>
<td>299</td>
</tr>
<tr>
<td>in transaction dump</td>
<td>297</td>
</tr>
<tr>
<td>registers at last EXEC command</td>
<td>298</td>
</tr>
<tr>
<td>RELATED dump code attribute</td>
<td>264, 265</td>
</tr>
<tr>
<td>Remote abend indicator</td>
<td>297</td>
</tr>
<tr>
<td>resource definition online (RDO)</td>
<td></td>
</tr>
<tr>
<td>ALTER mapset</td>
<td>5</td>
</tr>
<tr>
<td>ALTER program</td>
<td>5</td>
</tr>
<tr>
<td>ALTER transaction</td>
<td>5</td>
</tr>
<tr>
<td>DEFINE mapset</td>
<td>5</td>
</tr>
<tr>
<td>DEFINE program</td>
<td>5</td>
</tr>
<tr>
<td>DEFINE transaction</td>
<td>5</td>
</tr>
<tr>
<td>INSTALL option</td>
<td>5</td>
</tr>
<tr>
<td>resource names</td>
<td></td>
</tr>
<tr>
<td>&quot;CTLACB&quot;</td>
<td>117</td>
</tr>
<tr>
<td>AITM</td>
<td>114</td>
</tr>
<tr>
<td>ASYNRESP</td>
<td>114</td>
</tr>
<tr>
<td>ATCHMSUB</td>
<td>115</td>
</tr>
<tr>
<td>CDB2TIME</td>
<td>122</td>
</tr>
<tr>
<td>CEX2TERM</td>
<td>115</td>
</tr>
<tr>
<td>CHANGETERM</td>
<td>123, 133</td>
</tr>
<tr>
<td>CICS</td>
<td>118</td>
</tr>
<tr>
<td>CPI</td>
<td>115</td>
</tr>
<tr>
<td>CSASSI2</td>
<td>113</td>
</tr>
<tr>
<td>DB2START</td>
<td>122</td>
</tr>
<tr>
<td>DEBUGUSER</td>
<td>115</td>
</tr>
<tr>
<td>DCT</td>
<td>121, 135</td>
</tr>
<tr>
<td>DFH_STATE_TOKEN</td>
<td>122</td>
</tr>
<tr>
<td>DFHPTTW</td>
<td>115</td>
</tr>
<tr>
<td>DFHIPLT</td>
<td>119</td>
</tr>
<tr>
<td>DFHTEMP</td>
<td>122</td>
</tr>
<tr>
<td>DFHXMTA</td>
<td>118</td>
</tr>
<tr>
<td>DFHZARER</td>
<td>123</td>
</tr>
<tr>
<td>DFHZARL1</td>
<td>123</td>
</tr>
<tr>
<td>DFHZARL2</td>
<td>123, 134</td>
</tr>
<tr>
<td>DFHZARL3</td>
<td>123, 134</td>
</tr>
<tr>
<td>DFHZARL4</td>
<td>123</td>
</tr>
<tr>
<td>DFHZARQ1</td>
<td>123</td>
</tr>
<tr>
<td>DFHZARR1</td>
<td>123</td>
</tr>
<tr>
<td>DFHZCRO1</td>
<td>122, 133</td>
</tr>
<tr>
<td>DFHZDSP</td>
<td>121</td>
</tr>
<tr>
<td>DFHZEMW1</td>
<td>122, 133</td>
</tr>
<tr>
<td>DFHZERH1</td>
<td>123</td>
</tr>
<tr>
<td>DFHZERH2</td>
<td>123</td>
</tr>
<tr>
<td>DFHZERH3</td>
<td>123</td>
</tr>
<tr>
<td>DFHZERH4</td>
<td>123, 134</td>
</tr>
<tr>
<td>DFHZIS1</td>
<td>122</td>
</tr>
<tr>
<td>DFHZRAQ1</td>
<td>122, 133</td>
</tr>
<tr>
<td>DFHZRAR1</td>
<td>122, 133</td>
</tr>
<tr>
<td>DLCNTRL</td>
<td>113</td>
</tr>
<tr>
<td>DCONECT</td>
<td>113</td>
</tr>
<tr>
<td>DLSUSPND</td>
<td>114</td>
</tr>
<tr>
<td>DMWTQUEUE</td>
<td>113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource Names (continued)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS_NUDGE</td>
<td>121</td>
</tr>
<tr>
<td>DTCHMSUB</td>
<td>115</td>
</tr>
<tr>
<td>EARLYPLT</td>
<td>115</td>
</tr>
<tr>
<td>ECBTCP</td>
<td>113</td>
</tr>
<tr>
<td>EXCLOGER</td>
<td>114</td>
</tr>
<tr>
<td>EXECADDR</td>
<td>115</td>
</tr>
<tr>
<td>EXECSTRN</td>
<td>115</td>
</tr>
<tr>
<td>FDSESWR</td>
<td>115</td>
</tr>
<tr>
<td>FCSLDM</td>
<td>115</td>
</tr>
<tr>
<td>FCSREC</td>
<td>116</td>
</tr>
<tr>
<td>FCSRANGE</td>
<td>116</td>
</tr>
<tr>
<td>FCFLREC</td>
<td>116</td>
</tr>
<tr>
<td>FCFLMTL</td>
<td>116</td>
</tr>
<tr>
<td>file ID</td>
<td>115, 116, 117, 118</td>
</tr>
<tr>
<td>HVALUE</td>
<td>51</td>
</tr>
<tr>
<td>INITIAL</td>
<td>113</td>
</tr>
<tr>
<td>INQ_ECB</td>
<td>123</td>
</tr>
<tr>
<td>INQUIRE</td>
<td>133</td>
</tr>
<tr>
<td>inquiring during task waits</td>
<td>51</td>
</tr>
<tr>
<td>JOURNALS</td>
<td>116</td>
</tr>
<tr>
<td>JVM_POOL</td>
<td>115</td>
</tr>
<tr>
<td>KCADDR</td>
<td>116</td>
</tr>
<tr>
<td>KCSTRING</td>
<td>116</td>
</tr>
<tr>
<td>LATE_PLT</td>
<td>115</td>
</tr>
<tr>
<td>LG_MGRST</td>
<td>119</td>
</tr>
<tr>
<td>LIST</td>
<td>118, 123</td>
</tr>
<tr>
<td>LMQUEUE</td>
<td>93, 113</td>
</tr>
<tr>
<td>LOGSTRMS</td>
<td>116</td>
</tr>
<tr>
<td>LOT_ECB</td>
<td>114</td>
</tr>
<tr>
<td>MAXSOCKETS</td>
<td>120</td>
</tr>
<tr>
<td>message queue ID</td>
<td>122</td>
</tr>
<tr>
<td>MISCELANEOUS</td>
<td>120</td>
</tr>
<tr>
<td>module name</td>
<td>118</td>
</tr>
<tr>
<td>MRQUEUE</td>
<td>114</td>
</tr>
<tr>
<td>PSINQECB</td>
<td>123, 134</td>
</tr>
<tr>
<td>PSOP1ECB</td>
<td>123</td>
</tr>
<tr>
<td>PSOP2ECB</td>
<td>123, 134</td>
</tr>
<tr>
<td>PSUNBECB</td>
<td>123, 134</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>115</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>120</td>
</tr>
<tr>
<td>RZCBNOTI</td>
<td>119</td>
</tr>
<tr>
<td>SEND</td>
<td>120</td>
</tr>
<tr>
<td>SHUTECB</td>
<td>113</td>
</tr>
<tr>
<td>SINGLE</td>
<td>115, 118</td>
</tr>
<tr>
<td>SIPDMTEC</td>
<td>113</td>
</tr>
<tr>
<td>SMSYRE</td>
<td>121</td>
</tr>
<tr>
<td>SMSYSTEM</td>
<td>121</td>
</tr>
<tr>
<td>SO_LISTN</td>
<td>120</td>
</tr>
<tr>
<td>SO_LTEPTY</td>
<td>120</td>
</tr>
<tr>
<td>SO_LTERDC</td>
<td>121</td>
</tr>
<tr>
<td>SO_NOWORK</td>
<td>120</td>
</tr>
<tr>
<td>SOCLOSE</td>
<td>120</td>
</tr>
<tr>
<td>STATIC</td>
<td>117</td>
</tr>
<tr>
<td>STE</td>
<td>120</td>
</tr>
<tr>
<td>resource names</td>
<td>resource types</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>STP_DONE 113</td>
<td>EKCWAIT 115, 130</td>
</tr>
<tr>
<td>summary of possible values 111</td>
<td>ENF 115</td>
</tr>
<tr>
<td>SUSPEND 118</td>
<td>ENQUEUE 115, 116, 137</td>
</tr>
<tr>
<td>SUSPENDVALUE 51</td>
<td>ERDSA 67, 116</td>
</tr>
<tr>
<td>SYSIDNT/session ID 118</td>
<td>ESDSA 116</td>
</tr>
<tr>
<td>target transid 122</td>
<td>EUDSA 67, 116</td>
</tr>
<tr>
<td>TCLASS 121</td>
<td>FCAWAIT 116</td>
</tr>
<tr>
<td>TCTETI value 113</td>
<td>FCBFWAIT 83, 116</td>
</tr>
<tr>
<td>TCTVCECB 113</td>
<td>FCCAWAIT 83, 116</td>
</tr>
<tr>
<td>TDNQ 116</td>
<td>FCCFGR 84, 116</td>
</tr>
<tr>
<td>TERMINAL 119</td>
<td>FCCFQS 84, 116</td>
</tr>
<tr>
<td>terminal ID 118</td>
<td>FCCRWAIT 117</td>
</tr>
<tr>
<td>transient data queue name 113, 119, 121, 126</td>
<td>FCDWWAIT 84, 117</td>
</tr>
<tr>
<td>TSBUFFER 121</td>
<td>FCDSWAIT 84, 117</td>
</tr>
<tr>
<td>TSEXTEND 121</td>
<td>FCINWAIT 117</td>
</tr>
<tr>
<td>TSIO 121</td>
<td>FCIOWAIT 85, 117</td>
</tr>
<tr>
<td>TSNQ 116</td>
<td>FCIRWAIT 85, 117</td>
</tr>
<tr>
<td>TSQUEUE 122</td>
<td>FCPSWAIT 85, 117</td>
</tr>
<tr>
<td>TSTRING 122</td>
<td>FCRWAIT 86, 117</td>
</tr>
<tr>
<td>WTO 121</td>
<td>FCRBWAIT 86, 117</td>
</tr>
<tr>
<td>XM_HELD 119</td>
<td>FCRDWAIT 86, 117</td>
</tr>
<tr>
<td>ZC_ZGRP 121</td>
<td>FCRPWAIT 87, 117</td>
</tr>
<tr>
<td>ZGRPECB 113</td>
<td>FCRRWAIT 87, 117</td>
</tr>
<tr>
<td>ZSLSECB 122, 133</td>
<td>FCRVWAIT 87, 117</td>
</tr>
<tr>
<td>resource types</td>
<td>FCTISUSP 88, 117</td>
</tr>
<tr>
<td>ALLOCATE 113, 134</td>
<td>FCXCWAIT 88, 118</td>
</tr>
<tr>
<td>Any_MBCB 113, 138</td>
<td>FOREVER 118</td>
</tr>
<tr>
<td>Any_MRCB 113, 138</td>
<td>HTYPE 51</td>
</tr>
<tr>
<td>AP_INIT 113</td>
<td>ICEXPiry 118</td>
</tr>
<tr>
<td>AP_QUIES 113</td>
<td>ICGTWAIT 75, 118</td>
</tr>
<tr>
<td>AP_TERM 113</td>
<td>ICWAIT 75, 118</td>
</tr>
<tr>
<td>CCSTWAIT 113</td>
<td>inquiring during task waits 51</td>
</tr>
<tr>
<td>CDB2CONN 114, 126</td>
<td>IRLINK 57, 66, 118</td>
</tr>
<tr>
<td>CDB2RDYQ 114, 127</td>
<td>KC_ENQ 84, 130, 132</td>
</tr>
<tr>
<td>CDB2TCB 114, 127</td>
<td>KCCOMPAT 57, 118, 119, 120, 131</td>
</tr>
<tr>
<td>CDSA 67, 114</td>
<td>LATE_PLT 119</td>
</tr>
<tr>
<td>CFDTLRSW 114</td>
<td>LG_DEFER 119, 129</td>
</tr>
<tr>
<td>CFDTPOOL 114</td>
<td>LG_FORCE 119, 129</td>
</tr>
<tr>
<td>CFDTWAIT 114</td>
<td>LGDELALL 119, 129</td>
</tr>
<tr>
<td>CSNC 114</td>
<td>LGDRLAN 119, 129</td>
</tr>
<tr>
<td>DB2 114, 127</td>
<td>LGENDBLK 119, 130</td>
</tr>
<tr>
<td>DB2_INIT 114, 127</td>
<td>LGENDCRS 119, 130</td>
</tr>
<tr>
<td>DB2CDISC 114, 127</td>
<td>LGHARTBT 119</td>
</tr>
<tr>
<td>DB2EDISA 114, 128</td>
<td>LGREDBLK 119, 130</td>
</tr>
<tr>
<td>DBCTL 114, 128</td>
<td>LGREDCRS 119, 130</td>
</tr>
<tr>
<td>DBDXEOT 114</td>
<td>LGWRITE 119, 130</td>
</tr>
<tr>
<td>DBDXINT 114</td>
<td>MBCB_xxx 119, 138</td>
</tr>
<tr>
<td>DFHAIIIN 114</td>
<td>MRCB_xxx 119, 138</td>
</tr>
<tr>
<td>DFHCPIP 115</td>
<td>MXT 119</td>
</tr>
<tr>
<td>DFHPRIN 115</td>
<td>PIIS 119</td>
</tr>
<tr>
<td>DFHPTTW 115</td>
<td>PROGRAM 119, 120</td>
</tr>
<tr>
<td>DFHSIPLT 115</td>
<td>RCP_INIT 120</td>
</tr>
<tr>
<td>DISPATCH 115</td>
<td>RDSA 120</td>
</tr>
<tr>
<td>DMATTACH 115</td>
<td>SDSA 120</td>
</tr>
<tr>
<td>DTSKDEF 126</td>
<td>SMPRESOS 121</td>
</tr>
</tbody>
</table>
resource types (continued)
SOCKET 120
SODOMAIN 120, 121
SOSMVS 126
STP_TERM 121
SUCNSOLE 121
summary of possible values 111
SUSPENDTYPE 51
TCP_NORM 121
TCP_SHUT 121
TCTVCECB 121
TD_INIT 121, 135
TD_READ 121, 137
TDEPLOCK 121, 136
TDIPLOCK 121, 136
TIEXPIRY 121
TRANDEF 121
TSAUX 69, 121
TSBUFFER 70
TSEXTEND 70
TSIO 71
TSIOWAIT 122
TSPPOOL 71, 122
TSQUEUE 71
TSSHARED 71, 122
TSTRING 71
TSWBUF 71
UDSA 67, 122
USERWAIT 122, 130
WBALIAS 122
WEB_ECB 122
XRGETMSG 122
XRPUTMSG 140
ZC 122, 133
ZC_ZGRO 122, 133
ZC_ZGCH 123, 133
ZC_ZGIN 123, 133
ZC_ZGRP 123, 134
ZC_ZGUB 123, 134
ZCIOWAIT 123, 134
ZCZGET 123, 134
ZCZNAC 123, 134
ZXQOWAIT 123, 134
ZXSTWAIT 123

resources
DBCTL 128
definition errors 5
inquiring during task waits 51
locks 93
investigating waits 93
log manager 129
names 111
storage manager 67	
task control 130
temporary storage 69
types 111
RETAIN problem management system
APARs 315
data base 7, 313
problem management record 312
symptom keywords 7

RETAIN problem management system (continued)
using INFORMATION/ACCESS 7
RLS (record-level sharing)
taking SMSVSAM dumps 267

S
SAA (storage accounting area)
chains 194
overlays 194, 196, 199
SCRNSIZE attribute, PROFILE 178, 182
SDSA resource type 120
SDUMP macro 263
failure 263
retry on failure 263
SEND resource name 120
SENDSIZE attribute, TYPETERM 182
short-format trace 255
SHUTECB resource name 113
SINGLE resource name 118
SINGLE, resource name 115
SIPDMTEC resource name 113
SLIP trap, MVS 219
SMPSAM problems
taking RLS-related dumps 267
SMSYRE resource name 121
SMSYSTEM resource name 121
SO_LISTN resource name 120
SO_LTEPTY resource name 120
SO_LTERDC resource name 121
SO_NOWORK resource name 120
SOCKET resource type 120
SOCLOSE resource name 120
SODOMAIN resource type 120, 121
SOS (short on storage)
caused by looping code 14
potential cause of waits 68
SOSMVS resource name 115
SPCTR, system initialization parameter 237
SPCTRxx, system initialization parameter 237
specification exception 30
START PRINTER bit on write control character 185
statistics
autoinitiated tasks 14
file accesses 14
task control
number of times at MXT 163
use in investigating no task output 186
STE resource name 120
stealing TCBs 125
STGRCVY system initialization parameter 201
STNTR, system initialization parameter 237
STNTRxx, system initialization parameter 237
storage
consumption by looping tasks 68
fragmentation 69
task subpool summary 68
waits 67
fragmentation of free storage 69
too little free storage 68

362 CICS TS for z/OS: CICS Problem Determination Guide
storage chain checking
by CICS 194
forcing 197
storage freeze 303
storage manager domain (SM)
conditional storage requests 67
request for too much storage 68
suspend queue summary 68
trace levels 3 and 4 227
unconditional storage requests 67
waits 67
likely causes 67
storage recovery 201
storage violations
CHKSTRM option 197
CHKSTSK option 197
CICS detected 193, 194
CICS system dump 195, 196
exception trace entry 195, 196, 198
forcing storage chain checking 197
investigating 193
looking at the overlaying data 196
possible causes 200
programming errors 200
reason for invalid ECB address 131
symptoms 194
TIOA 194
undetected 193, 199
possible causes 199
user task storage element 194
using CSFE DEBUG 197
using trace 197, 199
STP_DONE resource name 113
STP_TERM resource type 121
STRFIELD option
CONVERSE command 185
SEND command 185
STRINGS parameter of FILE resource definition 85
structured fields 185
SUCNSOLE resource type 121
suppressing dumps
CICS dump table options 174
MVS Dump Analysis Elimination (DAE) 262
SUSPEND resource name 118
SUSPENDTYPE field 51
SUSPENDVALUE field 51
symbolic maps 191
symptom strings
in transaction dump 297
problem determination 38
RETAIN database search 297
RETAIN search keywords 38
symptoms
CICS has stopped running 8
CICS running slowly 9
incorrect output 11
keywords 7, 313
loops 12, 13
low priority tasks will not start 9
no output is obtained 9
poor performance 9, 12, 14, 161
symptoms (continued)
tasks do not complete 9
tasks in a wait state 12
tasks take a long time to complete 9
tasks take a long time to start 9
terminal activity is reduced 9
use in classifying problems 8
waits 12
SYS1.DUMP data set 263
SYSDUMP, system dump code attribute 175
SYSIDNT/session ID resource name 118
sysplex
MVS console commands 267
problem determination 265
resolving deadlocks 105
system busy symbol 13
system initialization
AUXTR parameter 247
AUXTRSW parameter 247
defining component tracing requirements 237
defining the tracing status 247
DUMP parameter 174, 261
DUMPDS parameter 262
DUMPSW parameter 262
DURETRY parameter 263
global suppression of CICS system dumps 261
GTFR parameter 247
INTTR parameter 247
setting transaction dump data set attributes 262
SPCTR parameter 237
SPCTRxx parameter 237
STNTR parameter 237
STNTRxx parameter 237
suppressing standard tracing 234
SYSTR parameter 234
TRTABSZ parameter 247
USERTR parameter 247
system loading, effect on performance 167
system task waits 112, 140
intentional 141
SYSTEM_DUMP, exit programming interface call 264
SYSTR, system initialization parameter 234

T

task control
waits 130
causes 131
failure of task to DEQUEUE on resource 132
invalid ECB address 131
resource type KCCOMPAT 131
unconditional ENQUEUE on single server resource 132
valid ECB address 132
task control area (TCA)
in transaction dump 298
system area 298
user area 298
task global table (TGT) 302
task termination
abnormal 3
task tracing

- precautions when choosing options 171
  special 233
  standard 233
  suppressed 233

tasks

- abnormal termination 3
- ATI, no output produced 185, 188
  looking at the AID chain 188
  looking at the ICE chain 188
- conversation state with terminal 65
- dispatch, suspend and resume cycle 162, 166
- dispatching priority 165
- error data 43
- exclusive control deadlock 89
- failure during MVS service call 43
- failure to complete 9, 10, 12
- failure to get attached to the dispatcher 162, 163
- failure to get attached to the transaction manager 161, 162
- failure to get initial dispatch 162, 165
- failure to start 9, 10, 12
- failure under the CICS RB 43
- identifying the AID 189
- identifying the ICE 188
- identifying, in remote region 66
- in a wait state 12
- in error 40
  identified in linkage stack 45
- information in kernel domain storage areas 41
- lock owning
  identifying a lock being waited on 93
- looping 143
  consumption of storage 68, 70
  identifying the limits 158
- MXT limit 163
- PSW at time of error 43
- reason for remaining on the AID chain 188
- registers at time of error 43
- runaway
  detection by MVS 43
  non-yielding loops 144
  storage report 42
  tight loops 144
- session state with VTAM 65
- slow running 10, 166
- subpool summary 68
- summary in kernel storage 40
- suspended 12
  inquiring on 12
  investigating 51
- task error information 41
- time-out interval 167
- tracing 171, 233
- transfer from ICE to AID chain 188
- waits 49
  CICS DB2 126
  DBCTL 128
  definition of wait state 49
  EDF 129
  log manager 129

Tasks (continued)

- waits (continued)
  maximum task conditions 96
  on locked resources 93
  online investigation 51
  stages in resolving wait problems 49
  storage manager 67
  suspension and resumption of tasks 111
  system 112
  task control 130
  techniques for investigating 51
  temporary storage 69
  user 112
  using the formatted CICS system dump 51
  using trace 51

TCAPCDSA field 302
TCLASS resource type 121
TCP_NORM resource type 121
TCP_SHUT resource type 121
TCSESUSF 188
TCTTE (terminal control table terminal entry)
  in transaction dump 300
TCTTE chain, in terminal waits 63
TCTVCECB resource name 113
TCTVCECB resource type 121
TD_INIT resource type 121
TD_READ resource type 121
TDEPLOCK resource type 121
TDIPLOCK resource type 121
TDNQ resource name 116
temporary storage
  conditional requests for auxiliary storage 69
  consumption by looping tasks 70
  current free space 69
  no task output 187
  repetitive records 14
  summary 70
  unconditional requests for auxiliary storage 69
  waits 69
  unallocated space close to exhaustion 69
terminal control program (TCP) 63
TERMINAL resource name 119
terminal tracing
  precautions when choosing options 171
  special 234
  standard 234
  suppressed 234
terminal waits
  access method 60
  autoinstall program not loaded 58
  CREATESESS(NO) in TYPETERM definition 64
  error action by TACP or NACP turned off 58
  finding the access method 59
  finding the type of terminal 59
  HANDLE CONDITION coded incorrectly 58
  interregion communication 66
  intersystem communication 59, 66
  ISMM access method 59
  multiregion operation 66
  identifying tasks in remote regions 66
  identifying the remote region 66
trace (continued)
incorrect output from (continued)
wrong tasks being traced 171
internal domain functions 227
interpreting 252, 256
user entries 258
interpreting user entries 258
investigating waits 51, 52
setting the tracing options 52
last command identification 301
last statement identification 301
level-1 227
level-2 227
level-3 227
levels 226, 228
logic of selectivity 235
master system trace flag 170, 234, 248
master user trace flag 248
MVS GTF 246
common destination 246
overview of different types 225
point id 252
points 226
location 227
program check and abend 231
repetitive output 14
storage manager trace levels 227
suspension and resumption of tasks 52
interpreting the trace table 52
use in investigating no task output 185
user 191
checking programming logic 191
user exception trace entries 228
VTAM buffer
description 232
destination 232
investigating logon rejection 178
terminal waits 66
TRANDEF resource type 121
TRANDUMP, transaction dump code attribute 175
transaction abends
abend code 25
documentation 26
interpretation 26
action of global trap/trace exit 306
AICA 26, 144
ASRA 26
ASRB 27
collecting the evidence 25
CSMT log messages 8
dump not made when expected 173
getting a transaction dump 25
investigating 25
last command identification 300
last statement identification 301
message 25
messages 3, 11
storage violation 196
system dump following 264
transaction dump following 264
worksheet 35
transaction deadlocks
in a sysplex 105
resolving 101
transaction dumps
abbreviated-format trace table 299
accompanying transaction abends 25
common system area 298
CSA 298
CSAOPFL 299
CWA 299
destination 262
DFHTACB 299, 301
dump not made on transaction abend 173
exec interface structure 298
EXEC interface user structure 298
execution key 297
extended-format trace table 299
following transaction abend 264
formatting 285
selectivity 285
in problem determination 261
interpretation 296
job log for DFHDU640 297
kernel stack entries 298
last command identification 300
last statement identification 301
locating program data 302
assembler program DFHEISTG storage 302
chain of PL/I DSAs 302
COBOL working storage 302
task global table 302
module index 300
optional features list 299
program information 300
program storage 300
PSW 297
registers 297
registers at last EXEC command 298
remote abend indicator 297
selective dumping of storage 264, 276
statistics 276
storage violation 195, 196
suppression for individual transactions 175, 261
symptom string 297
system EXEC interface block 298
task control area, system area 298
task control area, user area 298
TCTTE 300
transaction abend control block 299, 301
transaction dump code options 275
transaction dump codes 261
transaction storage 299
transaction work area 298
transaction list table (XLT) 109
transaction manager
failure of tasks to get initial attach 162
transaction routing 66
transaction storage
in transaction dump 299
transaction tracing
precautions when choosing options 171
transaction tracing  
(scontinued)

special 234
standard 234
suppressed 234
TRANSACTION_DUMP, exit programming interface
call 264
transactions

disabling 187
evidence that it ran 185
program control 186
program load 186
task attach 186
task dispatch 186
no output produced 184
ATI tasks 185, 188
disabling the transaction 187
explanatory messages 184
finding if the task ran 185
going at files 187
looking at temporary storage queues 187
looking at transient data queues 187
possible causes 184
START PRINTER bit on write control
character 185
task not in system 185
task still in system 185
testing the terminal status 184
using CECI 187
using execution diagnostic facility 186
using statistics 186, 187
using trace 185
wrong output produced 189
investigating 189
possible causes 189

transient data
extrapartition destinations 136
performance considerations 167
I/O buffers 138
intrapartition destinations 136
no task output 187
recoverable queues 136
VSAM I/O 138
VSAM strings 138
waits 135
during initialization 135
extrapartition 136
I/O buffer contention 138
I/O buffers all in use 138
intrapartition 136
resource names 135
resource types 135
VSAM I/O 138
VSAM strings all in use 138
translator
errors in output 6
traps 305
TRTABSZ, system initialization parameter 247
TSAUX resource type 69, 121
TSBUFFER resource name 70, 121
TSEXEND resource name 70, 121
TSIO resource name 71, 121
TSIOWAIT resource type 122
TSNQ resource name 116
TSPPOOL resource type 71, 122
TSQUEUE resource name 71, 122
TSHARED resource type 71, 122
TSTRING resource name 71, 122
TSWBUF resource name 71, 122
TYPETERM definition
ALTPAGE attribute 178, 182
ALTSCREEN attribute 178, 182
ATI status 189
CREATESESS(NO), cause of terminal waits 64
EXTEDEDDS attribute 178, 182
PAGESIZE attribute 182
SENDSIZE attribute 182
UCTRAN attribute 179

U

UCTRAN attribute 179
PROFILE definition 179
TYPETERM definition 179
UDSA resource type 122
uppercase characters 178
user task waits 112
user tracing
checking programming logic 191
exception trace entries 228
interpretation 258
USERTR, system initialization parameter 247
USERWAIT resource type 122

V

VARY NET,INACT command 110
VSAM
data buffers 83
exclusive control deadlock 89
exclusive control of control interval 88
I/O waits 85, 87
incorrect data read from file 182
index buffers 83
strings 85
transaction ID waits 88
waits
exclusive control deadlock 89
file state changes 84
for exclusive control of control interval 88
for VSAM transaction ID 88
I/O 85
record locking by CICS 91
VSAM buffer unavailable 83
VSAM string unavailable 85
VSAM READ SEQUENTIAL 89
VSAM READ UPDATE 89
VSAM WRITE DIRECT 90
VSAM WRITE SEQUENTIAL 90
VSMSTRING resource name 113
VTAM
process status 60
VTAM (continued)
    session state with task 65

W
WAIT symptom keyword 7
    waits
        alternate system 139
        CICS DB2 126
        DBCTL 128
        deadlock time-out interval 75
        definition 12, 49
        EDF 129
        enqueue 72
        FEPI 141
        file control 80
        IIOP 135
        interregion communication 66, 134
        intersystem communication 66, 134
        interval control 75
        investigating 49
        lock manager 93
        log manager 129
        maximum task conditions 96
        online investigation 51
            finding the resource 51
            program control 93
            stages in resolving 49
        storage manager 67
        suspension and resumption of tasks 111
        symptom keyword 7
        symptoms 12
        task control 130
        techniques for investigating 51
        temporary storage 69
        terminal 57
        transient data 135
            during initialization 135
            extrapartition 136
            I/O buffer contention 138
            I/O buffers all in use 138
            intrapartition 136
            VSAM I/O 138
            VSAM strings all in use 138
        using the formatted CICS system dump 51, 53
        using trace 51, 52
            setting the tracing options 52
        VTAM terminal control 133
        WBALIAS resource type 122
        WCC (write control character) 185
        WEB_ECB resource type 122
        working storage, COBOL programs 302
        write control character (WCC) 185
        WTO resource name 121

X
XDUREQ global user exit 261
XDUREQ, dump domain global user exit 174
XLT (transaction list table) 109
XM_HELD resource type 96

XRF errors
    failure of CAVM 140
XRF takeover
    CEDA in a wait state 140
    CESN in a wait state 140
    wait on resource type XRPUTMSG 140
XRGETMSG resource type 122
XRPUTMSG resource type 122, 140

Z
ZC resource type 122
ZC_ZGCH resource type 122
ZC_ZGIN resource type 123
ZC_ZGRP resource name 121
ZC_ZGRP resource type 123
ZC_ZGUB resource type 123
ZCIOWAIT 123
ZCZGET resource type 123
ZCZNAC resource type 123
ZGRPECB resource name 113
ZSLSECB resource name 122
ZXQOWAIT resource type 123
ZXSTWAIT resource type 123
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.
Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

• By mail, to this address:

  IBM United Kingdom Limited
  User Technologies Department (MP095)
  Hursley Park
  Winchester
  Hampshire
  SO21 2JN
  United Kingdom

• By fax:
  – From outside the U.K., after your international access code use 44–1962–816151
  – From within the U.K., use 01962–816151

• Electronically, use the appropriate network ID:
  – IBMLink: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

• The publication title and order number
• The topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.