

CICS Transaction Server for z/OS



# CICS System Definition Guide

*Version 3 Release 1*



CICS Transaction Server for z/OS



# CICS System Definition Guide

*Version 3 Release 1*

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 457.

This edition applies to Version 3 Release 1 of CICS Transaction Server for z/OS, program number 5655-M15, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

© Copyright IBM Corporation 1989, 2010.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Preface</b> . . . . .	xi
What this book is about . . . . .	xi
Who should read this book . . . . .	xi
What you need to know to understand this book . . . . .	xi
How to use this book . . . . .	xi
Notes on terminology . . . . .	xi
Book structure . . . . .	xi
<b>Summary of changes</b> . . . . .	xiii
Changes for CICS Transaction Server for z/OS, Version 3 Release 1 . . . . .	xiii
Changes for CICS Transaction Server for z/OS, Version 2 Release 3 . . . . .	xiii
Changes for CICS Transaction Server for z/OS, Version 2 Release 2 . . . . .	xiv
Changes for CICS Transaction Server for z/OS, Version 2 Release 1 . . . . .	xiv
Changes for the CICS Transaction Server for OS/390 Version 1 Release 3 edition . . . . .	xv
Changes for the CICS Transaction Server for OS/390 Version 1 Release 2 edition . . . . .	xvi
Changes for the CICS Transaction Server for OS/390 Version 1 Release 1 edition . . . . .	xvi

---

## Part 1. Defining resources . . . . . 1

<b>Chapter 1. Introducing resource definition</b> . . . . .	3
Using the CSD and control tables together . . . . .	5
<b>Chapter 2. Defining terminal resources</b> . . . . .	7
Defining VTAM terminals . . . . .	7
Defining CICS terminal resources to VTAM . . . . .	7
Defining terminal resources to CICS . . . . .	8
Defining the terminal shutdown time limit . . . . .	8
Choosing an appropriate value for TCSWAIT . . . . .	9
Limitations of the terminal shutdown time limit facility . . . . .	9
Defining remote TCAM terminals . . . . .	10
Defining sequential (BSAM) devices . . . . .	10
Using a sequential device with START requests . . . . .	12
Terminating . . . . .	12
Defining console devices . . . . .	14
Defining console devices to CICS . . . . .	14
VTAM persistent sessions considerations . . . . .	18
Unbinding sessions . . . . .	19
Sessions not retained . . . . .	19
XRF considerations . . . . .	20

---

## Part 2. Defining data sets . . . . . 21

<b>Chapter 3. Setting up CICS data sets</b> . . . . .	23
Data set naming conventions . . . . .	23
Multiple extents and multiple volumes . . . . .	26
Performance considerations of TS and TD buffers . . . . .	26
CICS-supplied jobs to create CICS data sets . . . . .	26
MVS system data sets used by CICS . . . . .	28
Setting up data sets for XRF . . . . .	29

Actively shared data sets . . . . .	29
Passively shared data sets . . . . .	30
Unique data sets . . . . .	30
Data set allocation . . . . .	30
Backup while open (BWO) of VSAM files . . . . .	31
Effect of disabling activity keypointing . . . . .	33
Restrictions on BWO. . . . .	33
XRF considerations . . . . .	33
Using storage management facilities . . . . .	33
Storage management subsystem (SMS) . . . . .	34
Data facility hierarchical storage manager (DFHSM) . . . . .	34
<b>Chapter 4. Setting up the temporary storage data set . . . . .</b>	<b>35</b>
Defining the temporary storage data set. . . . .	35
Using multiple extents and multiple volumes . . . . .	36
Space considerations . . . . .	36
Number of VSAM buffers and strings. . . . .	37
Job control statements for CICS execution. . . . .	37
XRF considerations . . . . .	37
Defining temporary storage pools for temporary storage data sharing . . . . .	38
Approximate storage calculations . . . . .	38
<b>Chapter 5. Setting up transient data destination data sets . . . . .</b>	<b>41</b>
Defining intrapartition data sets . . . . .	42
Job control statements to define the intrapartition data set . . . . .	43
Space considerations . . . . .	44
Number of VSAM buffers and strings. . . . .	44
Job control statements for CICS execution. . . . .	45
XRF considerations . . . . .	45
Defining extrapartition data sets. . . . .	46
The DFHCXRF data set . . . . .	46
XRF considerations . . . . .	47
<b>Chapter 6. Setting up CICS log streams . . . . .</b>	<b>49</b>
Defining CICS system logs . . . . .	49
Planning your CICS system log streams . . . . .	49
Defining CICS general logs . . . . .	51
Planning log streams for use by your user journals and autojournals . . . . .	51
Planning log streams for use by your forward recovery logs . . . . .	51
Planning log streams for use by your log of logs (DFHLGLOG) . . . . .	52
Naming journals . . . . .	53
System logs . . . . .	53
Forward recovery logs . . . . .	53
User journals . . . . .	54
Installing system log and journal names. . . . .	55
Defining JOURNALMODELS . . . . .	55
Mapping log streams. . . . .	56
Mapping of the system log stream. . . . .	56
Mapping of general log streams. . . . .	59
Using the Journal utility program, DFHJUP . . . . .	61
<b>Chapter 7. Setting up the CICS system definition data set . . . . .</b>	<b>63</b>
Creating a CSD . . . . .	63
Calculating CSD disk space . . . . .	64
Initializing the CSD . . . . .	65
Creating a larger CSD . . . . .	67

Defining CSD attributes . . . . .	67
Sharing the CSD in non-RLS mode . . . . .	68
Shared user access from the same CICS region . . . . .	68
Multiple users of the CSD within a CICS region (non-RLS) . . . . .	69
Sharing a CSD by CICS regions within a single MVS image (non-RLS) . . . . .	70
Sharing a CSD in a multi-MVS environment (non-RLS) . . . . .	71
Multiple users of one CSD across CICS or batch regions (non-RLS) . . . . .	71
Sharing the CSD between different releases of CICS . . . . .	72
Other factors restricting CSD access . . . . .	73
Sharing the CSD in RLS mode . . . . .	73
Differences in CSD management between RLS and non-RLS access . . . . .	74
Specifying file control attributes for the CSD . . . . .	75
Effect of RLS on the CSD batch utility DFHCSDUP . . . . .	75
Planning for backup and recovery . . . . .	76
Transaction backout during emergency restart . . . . .	78
Dynamic backout for transactions . . . . .	78
Other recovery considerations . . . . .	78
Logging RDO commands . . . . .	80
Making the CSD available to CICS . . . . .	81
Installing the RDO transactions . . . . .	82
Moving your CICS tables to the CSD . . . . .	82
Installing definitions for the Japanese language feature . . . . .	82
CSD XRF considerations . . . . .	82
<b>Chapter 8. Setting up and using catalog data sets . . . . .</b>	<b>85</b>
Defining the global catalog . . . . .	85
JCL to define and initialize the global catalog . . . . .	86
Space calculations . . . . .	89
Job control statement for CICS execution . . . . .	91
Defining the local catalog . . . . .	91
Information written to the local catalog . . . . .	91
Job control statements to define and initialize the local catalog . . . . .	93
Job control statement for CICS execution . . . . .	94
<b>Chapter 9. Setting up and using auxiliary trace data sets . . . . .</b>	<b>95</b>
Starting and controlling auxiliary trace . . . . .	95
Allocating auxiliary trace data sets . . . . .	96
Space calculations . . . . .	97
Job control statements for CICS execution . . . . .	97
XRF considerations . . . . .	97
Using the trace utility program (DFHTU640) . . . . .	98
<b>Chapter 10. Defining dump data sets . . . . .</b>	<b>99</b>
System dumps . . . . .	99
MVS SDUMP macro . . . . .	99
Suppressing system dumps that precede ASRx abends . . . . .	100
Processing system dumps . . . . .	101
The CICS transaction dump data sets . . . . .	101
Selecting the transaction dump data set at startup . . . . .	102
Job control statements to allocate dump data sets . . . . .	102
Copying disk dump data sets to tape . . . . .	103
Space calculations . . . . .	103
Job control statements for CICS execution . . . . .	103
<b>Chapter 11. Defining the CICS availability manager data sets . . . . .</b>	<b>105</b>
The XRF control data set . . . . .	105

JCL to define the XRF control data set . . . . .	106
Space calculations . . . . .	106
Job control statements for CICS execution . . . . .	107
The XRF message data set. . . . .	107
JCL to define the XRF message data set. . . . .	107
Space calculations . . . . .	108
Job control statement for CICS execution. . . . .	110
Security . . . . .	110
I/O error handling . . . . .	110
<b>Chapter 12. Defining user files . . . . .</b>	<b>111</b>
VSAM data sets . . . . .	112
VSAM bases and paths . . . . .	112
Loading empty VSAM data sets . . . . .	113
Reuse of data sets . . . . .	113
VSAM record-level sharing (RLS). . . . .	114
BDAM data sets . . . . .	116
Defining data sets to CICS . . . . .	118
Using JCL . . . . .	118
Opening VSAM or BDAM files . . . . .	120
Closing VSAM or BDAM files . . . . .	121
Closing files normally . . . . .	121
Closing files using the FORCE option . . . . .	121
XRF considerations. . . . .	121
CICS data tables. . . . .	123
Opening data tables . . . . .	123
Loading data tables. . . . .	123
Closing data tables . . . . .	123
XRF considerations. . . . .	124
Coupling facility data tables. . . . .	124
Comparison with user-maintained data tables . . . . .	124
Coupling facility data table models . . . . .	124
Coupling facility data table structures and servers . . . . .	125
Defining a coupling facility data table pool . . . . .	126
<b>Chapter 13. Defining the CDBM GROUP command data set . . . . .</b>	<b>131</b>
Job control statements for CICS execution . . . . .	132
Record layout in the CDBM GROUP command file . . . . .	132
<b>Chapter 14. Defining the CMAC messages data set . . . . .</b>	<b>135</b>
Job control statements to define and load the messages data set. . . . .	135
Job control statements for CICS execution . . . . .	136
<b>Chapter 15. Defining the EJB data sets . . . . .</b>	<b>137</b>
Defining EJB directory and object store data sets. . . . .	137
Determining the object store space requirements . . . . .	139
<b>Chapter 16. Defining the WS-AT data set . . . . .</b>	<b>141</b>
<b>Chapter 17. Setting up the debugging profiles data sets . . . . .</b>	<b>143</b>
Creating the debugging profiles data sets. . . . .	143
Defining the debugging profiles data sets as VSAM RLS files . . . . .	145
Defining the debugging profiles data sets as VSAM non-RLS files . . . . .	146
Defining the debugging profiles data sets as remote files . . . . .	147

#

#  
#

<b>Chapter 18. Specifying CICS system initialization parameters</b> . . . . .	151
System initialization parameters for open TCBS . . . . .	152
TCB considerations with UNIX System Services . . . . .	155
The implications of setting MAXPROCUSER too low . . . . .	156
Specifying DFHSIT macro parameters . . . . .	157
Initialization parameters that cannot be coded in the DFHSIT macro . . . . .	161
Defining CICS resource table and module keywords. . . . .	162
The system initialization parameter descriptions . . . . .	164
The default system initialization table . . . . .	263
Assembling the SIT. . . . .	270
Assembler errors from undefined keywords . . . . .	270
Selecting versions of CICS programs and tables . . . . .	270
Using an explicit level of function to select programs . . . . .	271
Excluding unwanted programs. . . . .	271
<b>Chapter 19. Processing system initialization parameters</b> . . . . .	273
Supplying system initialization parameters to CICS . . . . .	273
Using system initialization control keywords . . . . .	273
Processing the PARM parameter. . . . .	276
Processing the SYSIN data set . . . . .	276
Processing the console entries . . . . .	277
Controlling start and restart . . . . .	278
The role of the CICS catalogs . . . . .	278
The START system initialization parameter . . . . .	279
CICS startup and the VTAM session . . . . .	284
The CICS parameter manager domain. . . . .	285
End of CICS startup . . . . .	286
<b>Chapter 20. Defining the JVM options (JVM profiles and JVM properties files)</b> . . . . .	287
Rules for coding JVM profiles and JVM properties files. . . . .	287
Options in JVM profiles . . . . .	290
CICS-specific options . . . . .	295
Java standard options . . . . .	303
Java nonstandard options . . . . .	305
System properties for JVMs. . . . .	310
Standard system properties. . . . .	312
System properties specific to the IBM persistent reusable JVM. . . . .	316
The sample JVM profiles and JVM properties files . . . . .	321
<b>Chapter 21. CICS startup</b> . . . . .	337
Using the sample startup job stream . . . . .	338
A sample CICS startup job . . . . .	340
Storage requirements for a CICS region . . . . .	352
Storage protection . . . . .	354
The dynamic storage areas and associated storage cushions . . . . .	357
The sample statistics program, DFH0STAT . . . . .	360
A sample CICS startup procedure . . . . .	360
<b>Chapter 22. Preparing CICS for using debugging tools</b> . . . . .	363
Preparing your CICS region for debugging . . . . .	363

---

**Part 4. Initializing CICS data sharing servers** . . . . . 365

<b>Chapter 23. Authorized cross memory (AXM) system services</b> . . . . .	367
The authorized cross-memory (AXM) server environment . . . . .	367

<b>Chapter 24. Setting up and running a temporary storage server</b>	369
Overview of the temporary storage data sharing server	369
Security	370
Defining TS server regions	370
Sample startup job for a TS server	371
Queue server REGION parameter	371
TS queue server parameters	372
Primary parameters.	372
Automatic restart manager (ARM) parameters	373
List structure parameters	374
Debug trace parameters	374
Tuning parameters	375
Warning parameters	376
Automatic ALTER parameters	377
Queue server automatic ALTER processing	377
Shared TS queue server commands	378
DISPLAY and PRINT keywords	379
The CANCEL command options	380
Unloading and reloading queue pools	380
<b>Chapter 25. Setting up and running a coupling facility data table server</b>	383
Overview of a coupling facility data table server	383
Coupling facility data table structures and servers	383
Security	384
Defining and starting a coupling facility data table server region	384
The server region program, DFHCFMN	384
Coupling facility data table server parameters	386
Avoiding structure full conditions	394
Coupling facility data table server automatic structure alter	395
Controlling coupling facility data table server regions	396
The SET command options	397
DISPLAY and PRINT command options	398
The CANCEL command options	401
SETXCF commands	402
Deleting or emptying coupling facility data table pools	402
Unloading and reloading coupling facility data table pools.	402
<b>Chapter 26. Setting up and running a named counter server</b>	405
Named counter server overview	405
Named counter structures and servers.	406
Selecting a named counter server	406
Security	407
Defining a named counter options table	407
The options table parameters	407
Making an options table available to CICS	409
Defining a list structure	409
Defining and starting a named counter server region	411
The server region program, DFHNCMN	411
Named counter server parameters	412
Controlling named counter server regions	415
The SET command options	416
DISPLAY and PRINT command options	416
The CANCEL command options	418
Server response to XES events	418
Deleting or emptying named counter pools	418
Changing the size of named counter pools	418

	Unloading and reloading named counter pools . . . . .	419
	Unload JCL example . . . . .	421
	Reload JCL example . . . . .	421
	Dumping named counter pool list structures . . . . .	421
#	<b>Chapter 27. Coupling facility server operations</b> . . . . .	423
#	Monitoring coupling facility server messages . . . . .	423
#	Server messages . . . . .	423
#	AXM messages . . . . .	424
#	Coupling facility storage management . . . . .	424
#	Managing the pool structure . . . . .	425
#	Monitoring pool structure usage levels . . . . .	425
#	Operator messages reporting on pool structure usage . . . . .	425
#	Use of CFRM automatic ALTER to increase pool structure size. . . . .	426
#	Using system-managed rebuild to increase pool structure size . . . . .	426
#	Increasing the number of data lists . . . . .	426
#	Deleting or emptying the pool structure . . . . .	426
#	Server connection management . . . . .	427
#	Establishing server connections . . . . .	427
#	Terminating server connections . . . . .	427
#	Failed server connections . . . . .	428
#	Restarting a server . . . . .	428
	<b>Chapter 28. CICS server support for system-managed processes</b> . . . . .	431
	System-managed list structure rebuild . . . . .	431
	TS data sharing and CFDT servers . . . . .	432
	Named counter server. . . . .	432
	System-managed list structure duplexing . . . . .	433
	<b>Bibliography</b> . . . . .	435
	The CICS Transaction Server for z/OS library . . . . .	435
	The entitlement set . . . . .	435
	PDF-only books . . . . .	435
	Other CICS books . . . . .	437
	Books from related libraries. . . . .	437
	DATABASE 2 (DB2) . . . . .	437
	MVS . . . . .	437
	Java . . . . .	438
	Determining if a publication is current . . . . .	438
	<b>Accessibility</b> . . . . .	439
	<b>Index</b> . . . . .	441
	<b>Notices</b> . . . . .	457
	Trademarks. . . . .	458
	<b>Sending your comments to IBM</b> . . . . .	459



---

## Preface

---

### What this book is about

This book is intended to help you specify and install the system definitions and resources for a CICS® system. It contains guidance about the system definitions required to run a CICS system in an IBM® MVS™ environment.

---

### Who should read this book

This book is for system programmers responsible for specifying and installing the system definitions and resources for a CICS system.

---

### What you need to know to understand this book

You should have experience of the MVS operating system, and either have previous experience of CICS, or at least be familiar with the concepts and terminology. To understand the jobs required to install CICS resource definitions, you should be familiar with MVS job control language (JCL) and cataloged procedures.

---

### How to use this book

The parts and chapters of this book are self-contained. Use an individual part or chapter where it contains information about the particular task you are engaged in. For example, see Part 2, “Defining data sets,” on page 21 if your task is defining CICS data sets.

---

### Notes on terminology

“CICS” is used throughout this book to mean the CICS element of the IBM CICS Transaction Server for z/OS®.

“RACF” is used to mean the IBM Resource Access Control Facility (RACF®) or any other external security manager that provides equivalent function.

In the programming examples in this book, the dollar symbol (\$) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.

“MVS” is used throughout this book to mean the MVS operating system.

---

### Book structure

#### **Part1. Defining data sets ... pages “Defining of data sets” on page 21—“Job control statements for CICS execution” on page 136**

Describes the data sets needed by the various CICS facilities. Each chapter describes the facility, its function and usage, and the data sets required to implement it in a running CICS region.

**Part 2. System initialization ... pages “CICS system initialization” on page 149—“A sample CICS startup procedure” on page 360**

Describes the system initialization parameters that you can code to initialize a CICS region tailored for your installation.

**Part 3. Initializing CICS data sharing servers... pages Part 4, “Initializing CICS data sharing servers,” on page 365—“Dumping named counter pool list structures” on page 421**

Describes how to set up and start CICS data sharing servers.

---

## Summary of changes

This book is based on the CICS System Definition Guide for CICS Transaction Server for z/OS, Version 2 Release 1, SC34-5725-00. Changes from that edition are marked by vertical bars in the left margin.

This part lists briefly the changes that have been made for the following recent releases:

---

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

The more significant changes for this edition are:

- Technical changes :
  - New system initialization parameters:
  - Changed system initialization parameters:
  - Because of the removal of support for HPJ (hot-pooling) for Java™ program objects, the MAXHPTCBS system initialization parameter is removed.
- Structural changes:
  - The information previously given in the system initialization parameter list about open transaction environment TCBs has been moved to a separate section, “System initialization parameters for open TCBs” on page 152, and expanded.

---

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

The more significant changes for this edition are:

- Technical changes :
  - New chapter
    - Chapter 17, “Setting up the debugging profiles data sets,” on page 143
    - Chapter 22, “Preparing CICS for using debugging tools,” on page 363
  - New system initialization parameters:
    - “DEBUGTOOL” on page 184
    - “INFOCENTER” on page 204
    - “JVMPROFILEDIR” on page 207
    - “JVMCCPROFILE” on page 205
    - “JVMCCSIZE” on page 206
    - “JVMCCSTART” on page 206
    - “JVMLEVEL0TRACE” on page 207
    - “JVMLEVEL1TRACE” on page 207
    - “JVMLEVEL2TRACE” on page 207
    - “JVMUSERTRACE” on page 207
  - Changed system initialization parameters:
    - “MAXJVMTCBS” on page 210
  - There are extensive revisions to Chapter 20, “Defining the JVM options (JVM profiles and JVM properties files),” on page 287.
- Structural changes:
  - No major structural changes for this release.

---

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

The more significant changes for this edition are:

- Technical changes :
  - The use of Language Environment® is assumed in all system definition information. Runtime support is maintained for non-Language Environment conforming compilers and runtime libraries, but noapplication development guidance is given.
  - New chapter
    - Chapter 28, “CICS server support for system-managed processes,” on page 431
  - New system initialization parameters:
    - “AIBRIDGE” on page 164
    - “BRMAXKEEPTIME” on page 171
    - “EJBROLEPRFX” on page 190
    - “IIOPLISTENER” on page 204
    - “RSTSIGNOFF” on page 230
    - “RSTSIGNTIME” on page 231
    - “STATEOD” on page 241
    - “STATINT” on page 241
    - “XEJB” on page 258
- Changed system initialization parameters
  - “LGDFINT” on page 208
  - “XRFSOFF” on page 261
  - “XRFSTME” on page 262
- Structural changes:
  - The Appendix, *System initialization parameters grouped by functional area* has been removed for simplicity.

---

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

The more significant changes for this edition are:

- Moving material from CICS Transaction Server for OS/390® release 3
  - Moved to Resource Definition Guide
    - Part 1 Chapter 2 Defining resources in CICS control tables
  - Moved to Application Programming Guide
    - Part 1 Chapter 3 Installing map sets and partition sets
    - Part 1 Chapter 4 Installing application programs
  - Moved to Installation Guide
    - Part 1 Chapter 5 Defining DL/I support

- New chapters
  - Chapter 15, “Defining the EJB data sets,” on page 137
  - Chapter 20, “Defining the JVM options (JVM profiles and JVM properties files),” on page 287
- New system initialization parameters
  - “AUTODST” on page 168
  - “KEYRING” on page 208
  - “LGDFINT” on page 208
  - “MAXSOCKETS” on page 210
  - “SSLTCBS” on page 239

---

## Changes for the CICS Transaction Server for OS/390 Version 1 Release 3 edition

The main changes made to this book include:

- New chapters
  - Defining DB2® support
  - Defining sequence numbering resources
  - Defining and starting AXM system services
  - Starting a coupling facility data table server
  - Starting a named counter server
- Adding a CSECT to your map assembly
- More detail for:
  - The named counter application programming interface
  - KEYFILE
  - RRMS
  - SSDELAY
- New system initialization parameters
  - AICONS
  - DOCCODEPAGE
  - DSRTPGM
  - ENCRYPTION
  - FORCEQR
  - KEYFILE
  - MAXOPENTCBS
  - NCPLDFT
  - RRMS
  - RUWAPool
  - SSLDELAY
  - SSLTCBS
  - TCPIP
- Automatic restart
- Controlling named counter server regions
  - ARMREGISTERED
  - ARM
  - Cancel command

---

## Changes for the CICS Transaction Server for OS/390 Version 1 Release 2 edition

The major changes to CICS that affect CICS Transaction Server for OS/390, Version 1 Release 2 are:

- The MVS AUTODELETE and RETPD parameters, used to preserve data on the system log, and to manage the size of general logs, have been added to Chapter 6, “Setting up CICS log streams,” on page 49.
- The SYSLOG system initialization parameter, used to preserve data on the system log in CICS Transaction Server for z/OS, Version 2 Release 3, is obsolete in CICS Transaction Server for z/OS, Version 3 Release 1.
- The DB2 resource control table suffix is now specified on the DFHD2INI option of the INITPARM system initialization parameter.
- The chapter discussing the definition of DB2 support has been removed. All information about CICS DB2 is available in the *CICS DB2 Guide*.
- The following new system initialization parameters:
  - DB2CONN
  - DBCTLCON
  - WEB
  - WEBDELAY

---

## Changes for the CICS Transaction Server for OS/390 Version 1 Release 1 edition

The major changes to CICS Transaction Server for OS/390 Release 1 that affect this book are:

- The following new system initialization parameters:
  - SYDUMAX
  - TRDUMAX
  - CSDINTEG
  - CSDRLS
  - OFFSITE
  - RLS
  - SDTRAN
  - SYSLOG
  - TDINTRA
- Changes to the following system initialization parameters:
  - AILDELAY
  - AIRDELAY
  - DCT
  - START=INITIAL
- Resource definition online for transient data destinations.
- Resource definition online for journalmodels.
- An additional section in Chapter 4, “Setting up the temporary storage data set,” on page 35, discusses how to define temporary storage pools for temporary storage data sharing.
- An additional chapter to describe starting up a temporary storage server, together with information about all the initialization parameters that the TS server can use. See Chapter 24, “Setting up and running a temporary storage server,” on page 369.

---

## Part 1. Defining resources

After you have installed CICS, you need to define and install all the resources that your CICS regions need to run user transactions. This section of the book introduces resource definition concepts and requirements.

It consists of the following chapters:

- Chapter 1, “Introducing resource definition,” on page 3
- Chapter 2, “Defining terminal resources,” on page 7



---

## Chapter 1. Introducing resource definition

Before you can use CICS, you must supply it with information about the resources it should use, and how to use them. Some examples of resources are:

- Connections
- Databases
- Files
- Journals
- Journalmodels
- Programs
- Terminals
- Transactions
- Transient data queues (destinations)

Your CICS system has to know which resources to use, what their properties are, and how they interact with each other.

You supply this information to CICS by **resource definition**:

1. **Resource definition online (RDO)**: This method uses the CICS-supplied online transactions CEDA, CEDB, and CEDC. Definitions are stored on the CICS system definition (CSD) file, and installed into an active CICS system from the CSD file.
2. **DFHCSDUP offline utility**: This method also stores definitions in the CSD file. DFHCSDUP allows you to make changes to definitions in the CSD file by submitting a batch job offline.
3. **Automatic installation (autoinstall)**: This method minimizes the need for a large number of definitions, by dynamically creating new definitions based on a “model” definition that is provided by you.
4. **System programming, using the EXEC CICS CREATE commands**: You can use the EXEC CICS CREATE commands to create resources independently of the CSD file. For further information, see the *CICS System Programming Reference*.
5. **Macro definition**: You can use assembler macro source to define resources. Definitions are stored in assembled tables in a program library, and installed during CICS initialization.

The method you use depends on the resources you want to define. Table 1 on page 4 suggests some of the things you should consider when deciding which definition method to use when there is a choice.

Table 1. Methods of resource definition

Method	Description	Advantages	Disadvantages
RDO	This method uses the CEDA transaction, which allows you to define, alter, and install resources in a running CICS system.	RDO is used while CICS is running, so allows fast access to resource definitions.	Because CEDA operates on an active CICS system, care should be taken if it is used in a production system. Use some form of auditing as a control mechanism.
EXEC CICS CREATE system commands	This method allows you to add CICS resources to a CICS region without reference to the CSD file.	<ul style="list-style-type: none"> <li>• It enables configuration and installation of CICS resources for large numbers of CICS regions from a single management focal point.</li> <li>• It also allows you to write applications for administering the running CICS system.</li> </ul>	CREATE commands neither refer to nor record in the CSD file. The resulting definitions are lost on a cold start, and you cannot refer to them in a CEDA transaction.
DFHCSDUP	DFHCSDUP is an offline utility that allows you to define, list, and modify resources using a batch job. DFHCSDUP can be invoked as a batch program or from a user-written program running either in batch mode or under TSO.	<ul style="list-style-type: none"> <li>• You can modify or define a large number of resources in one job.</li> <li>• You can run DFHCSDUP against a non-recoverable CSD file while it is being shared between CICS regions using RLS access mode.</li> </ul>	<ul style="list-style-type: none"> <li>• You cannot install resources into an active CICS system.</li> <li>• You cannot make updates with DFHCSDUP against a recoverable CSD file that is being accessed in RLS mode.</li> </ul>
Autoinstall	This applies to VTAM® terminals, LU 6.2 sessions, journals, programs, mapsets, and partitionsets. You set up “model” definitions using either RDO or DFHCSDUP. CICS can then create and install new definitions for these resources dynamically, based on the models.	If you have large numbers of resources, much time is needed to define them, and if they are not all subsequently used, storage is also wasted for their definitions. Using autoinstall reduces this wasted time and storage.	You must spend some time initially setting up autoinstall in order to benefit from it.
Macro	<ul style="list-style-type: none"> <li>• Using this method, you code and assemble macroinstructions to define resources in the form of tables.</li> <li>• This method is available only for a limited number of resource types.</li> </ul>	None. However, this method must be used when no other methods are available.	<ul style="list-style-type: none"> <li>• You can change the definitions contained in the tables while CICS is running, but you must stop and restart CICS if you want it to use the changed tables.</li> <li>• You must do time-consuming assemblies to generate macro tables.</li> </ul>

For guidance on defining CICS resources, see the *CICS Resource Definition Guide*. For information about the DFHCSDUP utility, see the *CICS Operations and Utilities Guide*.

Resource definitions in the CSD are stored in **groups**. On a COLD or INITIAL start you specify the resource definitions that are required in a particular run of CICS by a **list** of groups. You can specify up to four lists of groups, using the GRPLIST=listname system initialization parameter, for installation during CICS

initialization. You can also use the CEDA INSTALL command to install a resource definition or group of definitions that are defined in the CSD dynamically on a running CICS region.

**Note:** The CSD is independent of the running CICS region, because when you install the definitions in the CICS region, CICS copies the information and keeps it in its own storage. Because CICS does this, you can change the CSD without interfering with the running CICS region. You can also change the definitions in the running CICS region by reinstalling them, or add more definitions by installing new ones.

You should limit read/write access to resource definitions in the CSD to a small number of people. To do this:

- Protect groups of resources by using the CEDA command LOCK.
- Protect the list of resource groups that is specified in the system initialization parameter GRPLIST by using the CEDA command LOCK.
- Use the CEDB transaction to create resource definitions, but not to INSTALL them.
- Use the CEDC transaction for read-only access to resource definitions.

CICS control tables contain resource definition records for resources not defined in the CSD. Using the CICS table assembly macro instructions creates the tables and their resource definitions. You must use macro instructions to define non-VTAM networks and terminals, non-VSAM files, databases, and resources for monitoring and system recovery. For more information about defining resources in CICS control tables, see *CICS Resource Definition Guide*.

---

## Using the CSD and control tables together

In the following cases, you can mix resources that are defined in the CSD with resources that are defined in control tables:

1. On an initial or cold start, you can mix file control resources that are defined in the CSD with those that were defined using DFHFCT macros. BDAM file definitions are loaded from the DFHFCT load module first, then the definitions for other types of files are loaded from the RDO groups specified in the GRPLIST system initialization parameter. When CICS is running, you can use CEDA commands to add more file resource definitions.
2. You can also mix terminal resource definitions for non-VTAM <sup>1</sup> terminals that are defined in a TCT with resource definitions for VTAM terminals that are defined using RDO.

However, avoid duplicate terminal IDs, because a TCT entry using the same terminal ID (TERMIDNT in the TCT) as a VTAM terminal in the CSD (TERMINAL name in the CSD), prevents CICS installing the VTAM definition.

---

1. Non-VTAM terminals. TCT entries can be for local or remote BSAM sequential devices, local or remote logical device codes (LDCs), and remote TCAM DCB terminals.



---

## Chapter 2. Defining terminal resources

This chapter describes how to define to CICS the terminals (and logical units) that it is to use, and how it is to use them. You define terminals to CICS in one of two ways, depending on the type of terminal access method you are using:

1. IBM ACF/VTAM terminals are defined in the CSD either explicitly, or by using model terminal definitions if you are using the CICS automatic installation facility (**autoinstall**). Using autoinstall, you leave it to CICS to install the terminal resource definition dynamically at logon time. CICS obtains the information needed to create a terminal entry from the `TERMINAL` and `TYPETERM` definitions recorded in the CSD. For guidance information about this process, see the *CICS Resource Definition Guide*.

You can add VTAM definitions to the CSD offline using the `DEFINE` command of the CICS utility program, `DFHCSDUP`, or online using the `CEDA DEFINE` command. If you want the terminal definitions installed during CICS initialization, you must add the names of the groups containing the definitions to a group list used during a cold start. Otherwise you can install a group of definitions using the `CEDA INSTALL GROUP(groupname)` command online. For details of the `GRPLIST` system initialization parameter, see Chapter 18, “Specifying CICS system initialization parameters,” on page 151.

Each terminal must also be defined to ACF/VTAM in a VTAM definition statement.

2. Non-VTAM terminals are defined in a terminal control table (TCT) using `DFHTCT` macros.

During CICS initialization, CICS loads the TCT specified by the TCT system initialization parameter, and those terminals defined in the TCT are installed as CICS resources. You must also make these terminals known to the operating system, and include a `DD` statement in the CICS startup job stream for each terminal.

If you are running CICS with XRF, see “XRF considerations” on page 20.

---

### Defining VTAM terminals

If your CICS system is to communicate with terminals or other systems using VTAM services, you must:

1. Define CICS to ACF/VTAM with an `APPL` statement in `SYS1.VTAMLST`. For more information about defining an `APPL` statement for CICS, see the *CICS Transaction Server for z/OS Installation Guide*.
2. Define to VTAM the terminal resources that CICS is to use. For more information about defining terminal resources to VTAM, see “Defining CICS terminal resources to VTAM.”
3. Define to CICS the terminal resources that it is to use. For more information about defining terminal resources to CICS, see “Defining terminal resources to CICS” on page 8.

### Defining CICS terminal resources to VTAM

Each terminal, or each logical unit (LU) in the case of SNA terminals, that CICS is to use must be defined to VTAM. The terminals can be defined as local or remote.

#### Local VTAM terminals

can be SNA terminals connected to a channel-attached cluster controller, or they can be non-SNA 3270 terminals connected through a local control unit.

### **Remote VTAM terminals**

are attached to an SNA cluster controller, which is connected through an SDLC line with a channel-attached communications controller. The communications controller may also be loaded with code to enable remote terminals to be connected to it by a binary synchronous (BSC) line.

You define terminals, controllers, and lines in VTAM tables<sup>2</sup> as nodes in the network. Each terminal, or each logical unit (LU) in the case of SNA terminals, must be defined in the VTAM tables with a VTAM node name that is unique throughout the VTAM domain.

If you are using VTAM 3.3 or later, you can define the AUTINSTMODEL name, printer, and alternate printer to VTAM by using VTAM MDLTAB and ASLTAB macros. These definitions are passed to CICS to select autoinstall models and printers.

For information about defining resources to VTAM, see the *ACF/VTAM Installation and Resource Definition* manual.

## **Defining terminal resources to CICS**

A given VTAM terminal (or logical unit) may be defined explicitly in the CICS system definition file (CSD), in which case it has a TERMINAL name, and a NETNAME (which is the same as the VTAM node name). Terminals defined in this way have terminal entries installed at CICS startup.

If a terminal does not have an explicit definition in the CSD, CICS can create and install a definition dynamically for the terminal when it logs on, using the CICS autoinstall facility. CICS can autoinstall terminals by reference to TYPETERM and model TERMINAL definitions created with the AUTINSTMODEL and AUTINSTNAME attributes. For information about TYPETERM and TERMINAL definitions, see the *CICS Resource Definition Guide*.

If you use autoinstall, you must ensure that the CICS resource definitions correctly match the VTAM resource definitions. For programming information about VTAM logmode definitions and their matching CICS autoinstall model definitions, see the *CICS Customization Guide*.

If you specify the system initialization parameter TCTUALOC=ANY, CICS stores the terminal control table user area (TCTUA) for VTAM terminals above the 16MB line if possible. (See page "TACTUALOC" on page 249 for more information about the TCTUALOC parameter.)

## **Defining the terminal shutdown time limit**

You can specify a time limit within which all VTAM terminals used by CICS must shut down, when CICS is shutting down. (This is to prevent a hung terminal stopping CICS shutting down.) You specify this time limit on the TCSWAIT system initialization parameter. You can also specify actions that CICS is to take, if the time limit is exceeded. You specify the actions on the TCSACTN system initialization parameter. More information about choosing appropriate values for TCSWAIT and TCSACTN is given in the following sections.

---

2. VTAM has tables describing the network of terminals with which it communicates. VTAM uses these tables to manage the flow of data between CICS and the terminals.

## Choosing an appropriate value for TCSWAIT

The value that you specify on the TCSWAIT system initialization parameter should be large enough so that under normal circumstances all VTAM terminals and connections shutdown in an orderly fashion. To help choose this value, consider using a value slightly larger than the elapsed time between the following two CICS terminal control shutdown messages:

```
DFHZC2305 Termination of VTAM sessions beginning
DFHZC2316 VTAM ACB is closed
```

**Note:** If you *do not* want a time limit (that is, you assume that all terminals never hang), specify the TCSWAIT=NO system initialization parameter.

### Specifying that CICS is only to report hung terminals

To report hung terminals and not attempt to force-close them specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters.

### Specifying that CICS is to force close all hung terminals

To attempt to force-close all hung terminals specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=UNBIND system initialization parameters.

### Specifying that CICS is to force close some hung terminals

To attempt to force-close some hung terminals, and only report others, specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters, and code a DFHZNEP routine that selects the required terminals and sets TWAOCN on for them.

### Specifying that CICS is to force close the VTAM ACB

To attempt to force-close the CICS VTAM ACB if there are any hung terminals, specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=FORCE system initialization parameters.

## Limitations of the terminal shutdown time limit facility

The following limitations apply to the terminal shutdown time limit:

- The terminal control shutdown time limit facility is only for VTAM terminals and VTAM intersystem connections.
- For all CICS-supported VTAM terminals, including LU Type 6.2 single-session APPC terminals (but excluding LU Type 6.1 connections and LU Type 6.2 parallel connections), the following facilities are provided:
  - The TCSWAIT-controlled shutdown timing mechanism
  - The TCSACTN- and DFHZNEP-controlled, optional, force close mechanism
  - The following messages:

```
DFHZC2350 Threshold exceeded. Sessions still active: ...
DFHZC2351 Terminal still active. Reason: ...
```
- For all VTAM intersystem connections, including both LU Type 6.1 connections and LU Type 6.2 parallel connection (but not LU Type 6.2 single-session APPC terminals), the following facilities are provided:
  - The TCSWAIT-controlled shutdown timing mechanism
  - The message: DFHZC2352 Connection still active
- The force-close action on a hung terminal (no quiesce protocol, issue VTAM CLSDST, send UNBIND to the terminal) only ATTEMPTS to shutdown the terminal; there is no guarantee that all terminals will shutdown in all

circumstances. To guarantee that all VTAM terminals will shutdown, and the VTAM ACB will be closed, you must specify TCSACTN=FORCE.

---

## Defining remote TCAM terminals

CICS TS 3.1 supports only *remote* TCAM terminals. Furthermore, it supports the DCB interface of ACF/TCAM (also known as the GET/PUT interface) but not the ACB interface. Thus, the only TCAM terminals you can define are those attached to a remote, pre-CICS TS 3.1, terminal-owning region by TCAM/DCB.

You cannot use RDO to define TCAM terminals. You must use resource definition macros. For information about using DFHTCT macros to define remote TCAM terminals, see the “Remote definitions for terminals for transaction routing” topic in the *CICS Resource Definition Guide*.

---

## Defining sequential (BSAM) devices

You can use a pair of input and output sequential data sets to simulate a terminal to CICS. For example, you can do this to test an application program before the intended terminal becomes available. You must code the following DFHTCT TYPE= macros:

```
DFHTCT TYPE=INITIAL,
        ACCMETH=(NONVTAM)    defining the access
                              method
```

(Define the following macro instructions contiguously.)

```
DFHTCT TYPE=SDSCI,
        DSCNAME=isadscn,    defining the input
        DDNAME=indd, ...    data set
DFHTCT TYPE=SDSCI,
        DSCNAME=osadscn,    defining the output
        DDNAME=outdd, ...   data set
DFHTCT TYPE=LINE,
        ISADSCN=isadscn,
        OSADSCN=osadscn, ...
DFHTCT TYPE=TERMINAL,
        TRMIDNT=name, ...
```

The two data sets defined by the DFHTCT TYPE=SDSCI macros simulate a CICS terminal known by the name specified in the TRMIDNT operand of the DFHTCT TYPE=TERMINAL macro. The DSCNAMEs of the input and output data sets must be specified in the ISADSCN and OSADSCN operands of the DFHTCT TYPE=LINE macro respectively.

You must code a DD statement for each sequential data set defined by an SDSCI macro. The DD name on the DD statement must be the same as the name coded on the DDNAME parameter (or, by default, on the DSCNAME parameter) of the SDSCI macro. For example, you could use the following DD statements for sequential input and output:

```
//CARDIN DD *,DCB=BLKSIZE=80
          .
          Statements containing valid transactions
          .
/*
//PRINTER DD SYSOUT=A,DCB=BLKSIZE=132
```

This example of an I/O combination simulates a terminal to a CICS application program. There is an example of the SDSCI statements supporting this

CARDIN/PRINTER combination in the copybook DFH\$TCTS, which is defined in the sample TCT, DFHTCT5\$. DFH\$TCTS is supplied in CICSTS31.CICS.SDFHSAMP. Input to the application program is submitted through the input stream (CARDIN), and output to the terminal is sent to the output stream (PRINTER). If the BLKSIZE parameter is defined in the TCT for the data set, you can omit it from the JCL. However, if it is not defined in the TCT, the block size defaults to 0, and if you also omit it from the DD statement for the data set, you get message IEC141I 013-34. There are other examples of DD statements for I/O sequential data sets in some of the CICS-supplied installation verification procedures. You can find them in CICSTS31.CICS.SDFHINST after installation.

You can also use two DASD data sets to simulate a terminal. You must code a DD statement for each data set defined by an SDSCI macro, and the DD name on the DD statement must be the name coded on the DDNAME (or DSCNAME) parameter of the SDSCI macro.

For example, you might code:

```
//DISKIN1 DD DSN=SIMULATD.TERMINAL.IN,  
//          UNIT=3380,DISP=OLD,VOL=SER=vol1id  
//DISKOT1 DD DSN=SIMULATD.TERMINAL.OUT,  
//          UNIT=3380,DISP=OLD,VOL=SER=vol1id
```

Input from this simulated terminal is read from the DISKIN1 data set. Output to the terminal is written to the DISKOT1 data set.

Each statement in the input file (from CARDIN or DISKIN1 in the examples used above), must end with a character representing X'E0'. The standard EBCDIC symbol for this end-of-data hexadecimal value is a backslash (\) character, and this is the character defined to CICS in the pregenerated system. You can redefine this for your installation on the EODI system initialization parameter; see Chapter 18, "Specifying CICS system initialization parameters," on page 151 for details.

## Using a sequential device with START requests

You can use a sequential device as the terminal specified on an EXEC CICS START REQUEST. This could be useful in situations where you need to associate the started task with a terminal, but where the termid specified does not need to represent a real terminal. For this purpose, define the sequential device as shown in Figure 1, and add the required DD statement to the CICS startup JCL.

```
*  
DFHTCT TYPE=INITIAL,SUFFIX=xx,           X  
          ACCMETH=(VTAM,NONVTAM)  
*  
DFHTCT TYPE=SDSCI,                       X  
          DEVICE=1403,                   X  
          DSCNAME=PRNT001  
*  
DFHTCT TYPE=LINE,                        X  
          ACCMETH=BSAM,                  X  
          INAREAL=80,                    X  
          TRMTYPE=CRLP,                  X  
          OSADSCN=PRNT001                X  
          DFHTCT TYPE=TERMINAL,          X  
          TRMIDNT=P001,                  X  
          ERRATT=NO,                      X  
          LPLEN=80,                       X  
          PGESIZE=(24,80),                X  
          TRMSTAT=RECEIVE  
*  
DFHTCT TYPE=FINAL
```

Figure 1. Example of TCT definitions needed to use BSAM device for START commands

Include the following DD statement in your CICS startup JCL to support the sequential device defined in Figure 1

```
//PRNT001 DD DUMMY
```

## Terminating

End-of-file does not terminate sequential input. You should use CESF GOODNIGHT as the last transaction, to close the device and terminate reading from the device. Otherwise, CICS invokes the terminal error program (DFHTEP), and issues the messages in Table 2 on page 13 at end-of-file on the sequential device.

Table 2. Warning messages if a sequential terminal is not closed

Message	Destination
DFHTC2507 <i>date time applid</i> Input event rejected return code zz {on line w/term at term} <i>termid</i> {, trans} <i>trandid</i> {, rel line=} <i>rr,time</i>	CSMT
DFHTC2500 <i>date time applid</i> {Line CU Terminal} out of service {Term W/Term} <i>termid</i>	CSMT

Use of CESF GOODNIGHT puts the sequential device into RECEIVE status and terminates reading from the device. However, if you close an input device in this way, the receive-only status is recorded in the warm keypoint at CICS shutdown. This means that the terminal is still in RECEIVE status in a subsequent warm start, and CICS does not then read the input file.

You can also use CESF LOGOFF to close the device and terminate reading from the device, but CICS still invokes DFHTEP to issue the messages in Table 2 at end-of-file. However, the device is left in TTI status, and is available for use when restarting CICS in a warm start.

If you want CICS to read from a sequential input data set, either during or following a warm start, you can choose one of the following methods:

- Close the input with CESF LOGOFF, and ignore the resultant messages. This leaves the terminal in TTI state, and CICS reads input automatically in the next startup.
- Do not close the input, and ignore the resultant messages. This leaves the terminal in TRANSCEIVE state, and CICS reads input automatically in the next startup.
- Close the input with CESF GOODNIGHT. This puts the sequential terminal into RECEIVE status and terminates reading from the terminal. In this case, it is recommended that you code a PLT program to change the status of the terminal to TRANSCEIVE (see below).
- Code a user program, to be invoked from the program list table (PLT), to issue the appropriate EXEC CICS INQUIRE and EXEC CICS SET commands for each sequential device that is required to process input. For example, use the following statement to establish the state of a sequential terminal:

```
EXEC CICS INQUIRE TERMINAL(termid) SERVSTATUS(cvda) TTISTATUS(cvda)
```

For each terminal where SERVSTATUS returns DFHVALUE(INSERVICE) and TTISTATUS returns DFHVALUE(NOTTI), set the terminal to TRANSCEIVE with the following statement:

```
EXEC CICS SET TERMINAL(termid) TTI
```

For programming information about the use of EXEC CICS INQUIRE and EXEC CICS SET commands, see the *CICS System Programming Reference* manual. For programming information about writing post initialization-phase programs, see the *CICS Customization Guide*.

If you use BSAM devices for testing purposes, the final transaction to close down CICS could be CEMT PERFORM SHUT. The receive-only status is recorded in the warm keypoint at CICS shutdown. This means that on a subsequent warm start of CICS, the terminal is still in RECEIVE status and CICS does not then read the input file.

If you use the CEMT PERFORM SHUT, perform a cold or initial start of the CICS region to read the input file. This assumes that recoverability of resources is not important to you.

---

## Defining console devices

You can operate CICS from a **console device** <sup>3</sup>.

You can use a terminal as both a system console and a CICS terminal. To enable this, you must define the terminal as a console in the CSD. (You cannot define consoles in the TCT.)

Suitably authorized TSO users can enter MODIFY commands from terminals connected to TSO. To enable this, define the TSO user as a console device in the CSD.

You can use each console device for normal operating system functions and to invoke CICS transactions. In particular, you can use the console device for CICS master terminal functions to control CICS terminals or to control several CICS regions in conjunction with multiregion operation. Consequently, you can be a master terminal operator for several CICS regions.

You can also use console devices to communicate with alternate CICS regions if you are using XRF. Such communication is limited to the CICS-supplied transaction, CEPT.

For more information about operating CICS from a console device, see the *CICS Operations and Utilities Guide*.

## Defining console devices to CICS

You can define console devices to CICS using either the DEFINE TERMINAL command of the DFHCSDUP utility, or the CEDA DEFINE TERMINAL command using RDO. Each console can be defined explicitly, or you can define autoinstall model definitions, and use the CICS terminal autoinstall facility for consoles to install consoles automatically.

If want to use the console autoinstall facility, specify AICONS=YESIAUTO as a system initialization parameter, and define TERMINAL model definitions that specify AUTINSTMODEL(YES) and the AUTINSTNAME attribute.

### System consoles

System consoles are defined to MVS in the SYS1.PARMLIB library, in a CONSOL $nn$  member, which defines attributes such as NAME, UNIT, and SYSTEM. The name is the most significant attribute, because it is the name that CICS uses to identify the console. The name is passed to CICS on an MVS MODIFY command. Note that although consoles also have a numeric identifier, this is allocated by MVS dynamically during IPL, and its use is not recommended for defining consoles to CICS.

For information about defining console devices to MVS, see the *OS/390 MVS Initialization and Tuning Reference*.

---

3. A console device can be a locally-attached system console, a TSO user defined as a console, or an automated process such as NetView®.

For information about defining MVS consoles to CICS, see “Defining MVS consoles to CICS.”

### **TSO users as consoles**

TSO users that issue commands to CICS, using either the TSO CONSOLE command or SDSF, do not require MVS definitions as consoles in the *CONSOLnn* member. MVS activates a console automatically using the user's TSO/E user ID as the console name

**Note:** The TSO user issuing the CONSOLE command can use the NAME option to specify a console name different from the TSO user ID.

To communicate with a CICS region from TSO or SDSF, you need to install a CICS console definition that specifies the TSO user ID (or the name specified on the console command) as the console name.

For information about the TSO CONSOLE command, see the *OS/390 TSO/E System Programming Command Reference*, SC28-1972

For information about defining TSO users to CICS, see “Defining TSO users as console devices” on page 16.

### **Defining MVS consoles to CICS**

To use an MVS console as a CICS master terminal, you either define it to CICS explicitly by a terminal definition entry in the CSD, or use the CICS console autoinstall facility.

| Each console you define is identified on the TERMINAL definition by the  
| CONSNAME(*name*) attribute. CICS no longer supports the CONSOLE(*number*)  
| attribute. Identify a console device attached to an MVS in a sysplex by its name,  
| using the CONSNAME attribute.

For an example of the DEFINE command required to define a console, see Figure 2 on page 16.

```

//DEFTERM JOB (accounting information),MSGCLASS=A,
//      MSGLEVEL=(1,1),CLASS=A,NOTIFY=userid
//CONSDEF EXEC PGM=DFHCSDUP
//STEPLIB DD DSN=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//DFHCSD DD DSN=CICSTS31.CICS.DFHCSD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
*
* Define a console for CICS
DEFINE TERMINAL(trmidnt) GROUP(grpname) TYPETERM(DFHCONS)
      CONSNAME(consname) DESCRIPTION(MVS CONSOLE consname)
*
* Define a TSO user as a console device for CICS
DEFINE TERMINAL(trmidnt) GROUP(grpname) TYPETERM(DFHCONS)
      CONSNAME(tsouser) DESCRIPTION(TSO USER tsouser)
      USERID(tsouser)
*
* Define an AUTOINSTALL model definition for a console device
DEFINE TERMINAL(autc) GROUP(grpname) TYPETERM(DFHCONS)
      CONSNAME(console) DESCRIPTION(Autoinstall model for a console)
      USERID(*FIRST) AUTINSTNAME(name) AUTINSTMODEL(YES)
*
ADD GROUP(grpname) LIST(yourlist)
*
LIST LIST(yourlist) OBJECTS
/*
//

```

Figure 2. Defining consoles and a TSO user in the CSD using DFHCSDUP

## Defining TSO users as console devices

You can define TSO users as console devices to CICS using either an explicitly-defined TERMINAL definition for each TSO user, or use the console autoinstall facility. To define a TSO user as a console, specify the console name used by the TSO user on the CONSNAME attribute of the DEFINE TERMINAL command. By default, the console name is the user's TSO user ID. You are recommended to define consoles to CICS with preset security by using the USERID operand, so that the TSO user does not have to sign on using the CESN transaction. Otherwise, the TSO user's CICS signon password is displayed when entered for the CESN transaction.

For an example of the DEFINE command required to define a TSO user, see Figure 2.

For information about defining consoles (or terminals) with preset security, see the *CICS RACF Security Guide*.

**Note:** Substitute your own values for the operands that are shown in italics in the DEFTERM job shown in Figure 2.

### AUTINSTMODEL(YES)

Specifies whether this TERMINAL definition can be used as a model for autoinstall purposes.

### AUTINSTNAME(*name*)

The name assigned to this autoinstall model definition, and by which the model is known to the autoinstall control program.

**CONSNAME**(*consname*)

A unique 8-character console name, which corresponds to the NAME parameter in the CONSOL*nn* PARMLIB member that defines the console, or matches the console name used by a TSO user.

You must define CONSNAME even for an autoinstall model.

**GROUP**(*grpname*)

A unique name for the group to which the console resource definition is to belong.

**LIST**(*yourlist*)

The startup group list containing the group in which you have defined the console definitions. If your new group list does not include the required CICS-supplied resources as well as your own, specify DFHLIST and *yourlist* on the GRPLIST system initialization parameter of your CICS startup job.

**TERMINAL**(*trmidnt* | *autc*)

A unique 4-character terminal identifier *trmidnt* as the name by which CICS is to identify the console in the TCT terminal entry (TCTTE), or a dummy name *autc* in the case of an autoinstall model definition.

**USERID**(*tsouser*)

The CICS preset security userid to be used to sign on this console device. For more information about preset terminal security, see the *CICS RACF Security Guide*.

If you have defined a console device in your CSD as CONSNAME(INTERNAL), you can use it to issue commands using MVS job control language. It is also used by authorized programs that use the MGCR macro to issue MVS commands.

Having defined the console devices in the CSD, ensure that their resource definitions are installed in the running CICS region. You can install the definitions in one of two ways, as follows:

1. Include the group list that contains the resource definitions on the GRPLIST system initialization parameter in the CICS startup job.
2. During CICS execution, install the console device group by using the RDO command CEDA INSTALL GROUP(*groupname*), where *groupname* is the name of the resource group containing the console device definitions.

DFHLIST, the CICS-defined group list created when you initialize the CSD with the DFHCSDUP INITIALIZE command, does not include any resource definitions for console devices. However, the CSD is initialized with 2 groups that contain console definitions:

**DFH\$CNSL**

This group contains definitions for three consoles. The group is intended for use with the installation verification procedures and the CICS-supplied sample programs. You can add this to your own group list, and alter the definitions to define your own console devices.

**DFHTERM**

This group contains a single definition of an autoinstall model definition for an MVS console.

If you decide to create new terminal definitions for your console devices, you can specify the CICS-supplied TYPETERM definition, DFHCONS, on the TYPETERM(*name*) parameter. This TYPETERM definition for console devices is generated in the group DFHTYPE when you initialize the CSD.

For information about TERMINAL definitions, see the *CICS Resource Definition Guide*.

---

## VTAM persistent sessions considerations

Persistent sessions support improves the availability of CICS. It benefits from VTAM 3.4.1 persistent LU-LU session improvements to provide restart-in-place of a failed CICS without rebinding. VTAM 4.3 introduced multi node persistent sessions which allows for VTAM failures and MVS failures as well as CICS failures. See the VTAM Network Implementation Guide for how to set up VTAM for MNPS and how and when it can be used.

CICS support of persistent sessions includes the support of all LU-LU sessions except LU0 pipeline and LU6.1 sessions. CICS determines for how long the sessions should be retained from the PSDINT system initialization parameter. This is a user-defined time interval. If a failed CICS is restarted within this time, it can use the retained sessions immediately—there is no need for network flows to rebind them.

You can change the interval using the CEMT SET VTAM command, or the EXEC CICS SET VTAM command, but the changed interval is not stored in the CICS global catalog, and therefore is not restored in an emergency restart.

If CICS is terminated through CEMT PERFORM SHUTDOWN IMMEDIATE, or if CICS fails, its sessions are placed in “recovery pending” state. During emergency restart, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an “in session” state. This happens when CICS opens its VTAM ACB.

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions are similar to that for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintain the bound session when possible but there are cases where it is necessary to unbind and rebind the sessions, for example, where CICS fails during a session resynchronization.

The end user of a terminal sees different symptoms of a CICS failure following a restart, depending on whether VTAM persistent sessions, or XRF, are in use:

- If CICS is running without VTAM persistent sessions or XRF, and fails, the user sees the VTAM logon panel followed by the “good morning” message (if AUTOCONNECT(YES) is specified for the TYPETERM resource definition).
- If CICS does have persistent sessions support and fails, the user perception is that CICS is “hanging”: the screen on display at the time of the failure remains until persistent session recovery is complete. After a successful CICS emergency restart, the recovery options defined for the terminals or sessions take effect. The recovery options are specified on the RECOVOPTION parameter of the TYPETERM resource definition. If you specify SYSDEFAULT as the value for RECOVOPTION, the terminal user can clear the screen and continue to enter CICS transids. If you specify MESSAGE as the RECOVNOTIFY attribute of the TYPETERM resource definition, the user is notified of the successful recovery.

## Unbinding sessions

Sessions held by VTAM in a recovery pending state are not always reestablished by CICS. CICS (or VTAM) unbinds recovery pending sessions in the following situations:

- If CICS does not restart within the specified persistent session delay interval.
- If you perform a COLD start after a CICS failure.
- If CICS restarts with XRF=YES (when the failed CICS was running with XRF=NO).
- If CICS cannot find a terminal control table terminal entry (TCTTE) for a session (for example, because the terminal was autoinstalled with AIRDELAY=0 specified).
- If a terminal or session is defined with the recovery option (RECOVOPT) set to UNCONDREL or NONE.
- A connection is defined with the persistent session recovery option (PSRECOVERY) set to NONE.

In all these situations, the sessions are unbound, and the result is as if CICS has restarted following a failure without VTAM persistent session support.

There are some other situations where APPC sessions are unbound. For example, if a bind was in progress at the time of the failure, sessions are unbound.

## Sessions not retained

There are some circumstances in which VTAM does not retain LU-LU sessions:

- VTAM does not retain sessions after a VTAM, MVS, or processor (CPC) failure.
- VTAM does not retain CICS sessions if you close VTAM with any of the following CICS commands:
  - SET VTAM FORCECLOSE
  - SET VTAM IMMCLOSE
  - SET VTAM CLOSED
- VTAM does not retain CICS sessions if you close the CICS node with the VTAM command VARY NET INACT ID=applid.
- VTAM does not retain CICS sessions if you perform a normal CICS shutdown (with a PERFORM SHUTDOWN command).

Without persistent sessions support, all sessions existing on a CICS system are lost when that CICS system fails. In any subsequent restart of CICS, the rebinding of sessions that existed before the failure depends on the terminal's AUTOCONNECT option. If AUTOCONNECT is specified for a terminal, the user of that terminal waits until the GMTRAN transaction has run before being able to continue working. If AUTOCONNECT is not specified for a terminal, the user of that terminal has no way of knowing (unless told by support staff) when CICS is operational again unless the user tries to log on. In either case, users are disconnected from CICS and need to reestablish a session, or sessions, to regain their working environment.

For CICS persistent session support, you need the VTAM persistent LU-LU session enhancements in VTAM 3.4.1 or later. CICS Transaction Server for z/OS, Version 3 Release 1 functions with releases of VTAM earlier than 3.4.1, but in the earlier releases sessions are not retained in a bound state in the event of a CICS failure.

---

## XRF considerations

If you intend operating CICS with the extended recovery facility (XRF), there are some more things you must consider when setting up your terminal network. For example, in an XRF environment, SNA VTAM terminals can be XRF-capable. This means that, if you have specified appropriate options in the TYPETERM definitions, XRF backup sessions can be established in parallel with the active sessions. (For guidance about defining extended recovery attributes for terminals, see the *CICS Resource Definition Guide*.)

The ability of a terminal to receive this XRF support is not determined by CICS, but by the terminal connection to CICS through ACF/NCP and ACF/VTAM. CICS gives each terminal the best support possible, based on the parameters passed to it from VTAM when the terminal logs on to CICS.

There are extra terminal definition keywords, that enable you to control the manner in which the XRF-capable terminals are supported by CICS.

Non-XRF-capable terminals must have their sessions reestablished to the new active CICS region after a takeover. How you do this depends on the network and the XRF configuration.

For further information about XRF-capable terminals, and non-XRF-capable terminals, see the *CICS/ESA 3.3 XRF Guide*.

---

## Part 2. Defining data sets

This part of the book describes the CICS system data sets needed to support various CICS facilities, such as temporary storage, transient data, transaction dump, trace, and so on. Some of these data sets are optional, others are required in order to run CICS. If a data set needs preformatting, a job that you can use for this purpose is shown.

The CICS facilities and their data sets are dealt with in the following sections:

- Chapter 3, “Setting up CICS data sets,” on page 23
- Chapter 4, “Setting up the temporary storage data set,” on page 35
- Chapter 5, “Setting up transient data destination data sets,” on page 41
- Chapter 6, “Setting up CICS log streams,” on page 49
- Chapter 7, “Setting up the CICS system definition data set,” on page 63
- Chapter 8, “Setting up and using catalog data sets,” on page 85
- Chapter 9, “Setting up and using auxiliary trace data sets,” on page 95
- Chapter 10, “Defining dump data sets,” on page 99
- Chapter 11, “Defining the CICS availability manager data sets,” on page 105
- Chapter 12, “Defining user files,” on page 111
- Chapter 13, “Defining the CDBM GROUP command data set,” on page 131
- Chapter 14, “Defining the CMAC messages data set,” on page 135
- Chapter 15, “Defining the EJB data sets,” on page 137
- Chapter 16, “Defining the WS-AT data set,” on page 141
- Chapter 17, “Setting up the debugging profiles data sets,” on page 143
- Chapter 28, “CICS server support for system-managed processes,” on page 431



---

## Chapter 3. Setting up CICS data sets

This chapter shows you how to define the data sets you need to run CICS. Some of these data sets are mandatory, whereas others are needed only if you are using the corresponding facilities. You may also need to provide data set definitions for user files, DL/I databases, and terminals other than VTAM terminals.

Space calculations are given so that you can calculate the space to allocate to the data sets, and the data definition statements to define them to the running CICS region.

CICS utility programs provided for postprocessing of the data sets are described in the *CICS Operations and Utilities Guide*.

Before you start setting up your CICS data sets, review the CICS facilities you need, and their data set requirements. You then have to:

- Define and catalog the data sets that are used by CICS during execution.
- If necessary, initialize or preformat the data sets for use during CICS execution.
- RACF-protect the data sets to suit your security requirements.
- Include in the CICS startup job stream DD statements for the required data sets, but note that DD statements are **not** needed for the following:
  - User files for which you are using CICS dynamic allocation facilities
  - CICS system data sets that are managed by CICS file control and for which you are using CICS dynamic allocation facilities
  - DL/I databases you are accessing through CICS remote DL/I support or DBCTL

For more information about user file definitions, see Chapter 12, “Defining user files,” on page 111.

Table 3 on page 24 summarizes the CICS data sets and their characteristics.

---

### Data set naming conventions

There are no restrictions on the data set names you choose for CICS data sets, other than MVS constraints. In the examples in this book, CICSSTS31.CICS is used as the high-level qualifier, and the DD name as the lowest level. If you are running multiple CICS regions, and especially if you are running CICS with XRF, you can use the CICS APPLID as a second level qualifier.

You are recommended to use the *CTGI* naming convention, as described in the *OS/390 Parallel Sysplex Application Migration* manual, GC28-1863.<sup>4</sup>For example, if CICSHTH1 is the APPLID, the data set name for the CSD would be:

```
DFHCSD DD DSN=CICSTS31.CICS.CICSHTH1.DFHCSD,DISP=SHR
```

---

4. The *CTGI* naming convention is a recommended example of a naming convention that you can use for CICS 4-character names, and is based on the 4-character *CTGI* symbol, where:

- C identifies an entire CICSplex
- T identifies the type of region
- G identifies a group of regions
- I identifies iterations of regions within a group

Where names are allowed to be up to eight characters long, as for CICS APPLIDs, the general recommendation is that the letters CICS are used for the first four characters, particularly for production regions.

If the data set is shared between an active CICS region and an alternate CICS region, use the generic APPLID, but if the data set is unique to either the active or the alternate CICS region, use the specific APPLID. For information about actively and passively shared data sets, see “Setting up data sets for XRF” on page 29.

Table 3. Summary of CICS data sets

Data set	DDNAME	Block or CI size (bytes)	Record format	Data set organization	Other comments
Auxiliary trace (See page “Setting up and using auxiliary trace data sets” on page 95)	DFHAUXT DFHBUXT	4096	F	Sequential	<b>3</b> See page 28 for information about GTF.
BTS local request queue. (See the <i>CICS Business Transaction Services</i> manual)	DFHLRQ	1024 and 2560	VB	VSAM KSDS	Required <i>even if you do not use BTS facilities</i> . BTS is described in the <i>CICS Business Transaction Services</i> manual.
CAVM control (See page “Defining the CICS availability manager data sets” on page 105)	DFHXRCTL	4096 minimum	<b>1</b>	VSAM ESDS	Required if running CICS with XRF.
CAVM message (See page “Defining the CICS availability manager data sets” on page 105)	DFHXRMSG	4096 minimum	<b>1</b>	VSAM ESDS	Required if running CICS with XRF.
Catalogs (See page “Setting up and using catalog data sets” on page 85)	DFHGCD DFHLCD	8192 & 2048	VB	VSAM KSDS	Both data sets must be initialized before use. <b>2</b>
CDBM group command (See page “Defining the CDBM GROUP command data set” on page 131)	DFHDBFK	8192	VB	VSAM KSDS	Required <i>only if you intend to use this function</i> .
CSD (See page “Setting up the CICS system definition data set” on page 63)	DFHCSD	8192	VB	VSAM KSDS	—
DJAR mapping file (See page Chapter 15, “Defining the EJB data sets,” on page 137)	DFHADJM	8192	F	VSAM KSDS	Required for CICS EJB development deployment tool only. This data set must not be shared.
Dump (See page “Defining dump data sets” on page 99)	DFHDMPA DFHDMPB	32 760 (tape) or 1 track (DASD)	V	Sequential	For CICS transaction dumps only; see page 28 for information about system dumps.
EJB directory (See page Chapter 15, “Defining the EJB data sets,” on page 137)	DFHEJDIR	1024	F	VSAM KSDS	Required only for use by the regions (listeners and AORs) in a logical EJB server. This data set must be shared by all the regions in the EJB server.

Table 3. Summary of CICS data sets (continued)

Data set	DDNAME	Block or CI size (bytes)	Record format	Data set organization	Other comments
EJB object store (See page Chapter 15, “Defining the EJB data sets,” on page 137)	DFHEJOS	8192	F	VSAM KSDS	Required only for use by the regions (listeners and AORs) in a logical EJB server. This data set must be shared by all the regions in the EJB server.
Messages & codes (See page “Defining the CMAC messages data set” on page 135)	DFHCMACD	—	V	VSAM KSDS	Can be created and loaded by the DFHCMACI job.
Temporary storage (See page “Setting up the temporary storage data set” on page 35)	DFHTEMP	See page 36	<b>1</b>	VSAM ESDS	—
Transient data extrapartition (See page 42) <b>4</b>	From the DDNAME option in the resource definition	From the BLOCKSIZE option in the resource definition	From the RECORD FORMAT option in the resource definition	Sequential	The TYPE=SDSCI macro referred to has the same DSCNAME parameter as the TYPE=EXTRA macro
Transient data intrapartition (See page 46)	DFHINTRA	See page “Space considerations of intrapartition data sets” on page 44.	<b>1</b>	VSAM ESDS	—

**Notes:**

**1** These data sets use control interval (CI) processing and therefore the record format is not relevant.

**2** DFHGCD is the CICS global catalog data set, and in an XRF environment it is passively shared between the active and the alternate CICS regions. DFHLCD is the CICS local catalog data set, and this is a unique data set; each CICS region must have its own local catalog. See “Setting up data sets for XRF” on page 29 for an explanation of actively and passively shared data sets in an XRF environment.

**3** The CICS utility program, DFHTU640, prints and formats auxiliary trace data. For information about this CICS utility program, see the *CICS Operations and Utilities Guide*.

**4** You do not have to specify all the data sets associated with extrapartition transient data queues in the CICS JCL this is because of the introduction of dynamic allocation of extrapartition transient data queue data sets. See the *CICS Resource Definition Guide* for more information.

## Multiple extents and multiple volumes

You can define a temporary storage data set, or a transient data destination data set, as a single extent defined on a single volume. That data set must be big enough to hold all your data. Instead of defining one data set, which to cater for exceptional cases might have to be much larger than your average, you can define:

- Multiple extents on one volume
- One extent on each of multiple volumes
- Multiple extents on multiple volumes

When you define more than one extent, CICS uses the extra extents only when the primary extent is full. You could make your primary extent large enough to meet average demand, and then have smaller secondary extents for overflow. In this way, you save space until it is needed. When each extra extent becomes full, VSAM creates another. VSAM continues to create extra extents, as needed, up to a maximum of 123 extents. The use of multiple volumes has no effect on this limit.

To allocate additional extents in the same volume, code a secondary extent operand on the RECORDS parameter:

```
RECORDS(primary,secondary)
```

To use single extents on multiple volumes, code:

```
RECORDS(primary) -  
VOLUMES(volume1,volume2,volume3,....)
```

For multiple extents on multiple volumes, combine both primary and secondary RECORDS operands with multiple VOLUMES operands:

```
RECORDS(primary,secondary) -  
VOLUMES(volume1,volume2,volume3,....)
```

If a particular volume causes performance bottlenecks, try single extents on multiple volumes.

Multiple extents over multiple volumes should be used if there is a probability that a volume will exhaust its free space before VSAM reaches its limit on extra extents. If this occurs, VSAM continues to create extra extents on the next volume in the list.

## Performance considerations of TS and TD buffers

When specifying the number of buffers for temporary storage and transient data, you should consider the following possible performance impact:

- Using a large number of buffers means that for non-recoverable queues processing can often be performed without calling VSAM. This improves CICS performance. However, at shutdown all non-empty buffers have to be flushed sequentially which can take a long time.

---

## CICS-supplied jobs to create CICS data sets

CICS supplies the following jobs that you can use to create the CICS data sets.

<b>Job</b>	<b>Function</b>
<b>DFHCOMDS</b>	Deletes and re-creates data sets common to all CICS regions. (See Table 4 on page 27.)
<b>DFHDEFDS</b>	Deletes and re-creates copies of data sets used only by one CICS region. (See Table 5 on page 27.) You run a separate copy of this job to create the data sets for each CICS region.

**DFHCMACI** Deletes and re-creates the CICS messages data set, DFHCMACD, and loads it with the data from the CICS-supplied file, DFHCMACD, in the CICSTS31.CICS.SDFHMSG target library.

When you ran the DFHISTAR job as part of the CICS installation or post-installation tasks, these jobs were tailored to your environment and stored in the library that you specified on the LIB parameter of the DFHISTAR job (by default, CICSTS31.XDFHINST). If you have not yet run DFHISTAR, you should do so before running any of the CICS post-installation jobs.

You can generate several copies of these jobs by rerunning the DFHISTAR job, selecting the jobs that you want to copy. To generate new copies of these jobs, edit the DFHISTAR job to specify new values for the DSINFO and SELECT parameters. Only those jobs that you name by the SELECT parameter are regenerated.

For information about these jobs and about generating new versions of them, see the *CICS Transaction Server for z/OS Installation Guide*.

*Table 4. CICS data sets created by the DFHCOMDS job*

DFHCSD	CICS region definition data set
SYSIN	SYSIN data set

*Table 5. CICS data sets created by the DFHDEFDS job*

DFHADJM	Application deployment DJAR mapping file
DFHAUXT	non-VSAM auxiliary trace (A) data set
DFHBUXT	non-VSAM auxiliary trace (B) data set
DFHDMPA	non-VSAM dump (A) data set
DFHDMPB	non-VSAM dump (B) data set
DFHEJDIR	EJB directory
DFHEJOS	EJB object store
DFHGCD	CICS global catalog
DFHINTRA	intrapartition transient data set
DFHLCD	CICS local catalog
DFHLRQ	BTS local request queue
DFHTEMP	temporary storage data set
DFHXRCTL	XRF control data set
DFHXRMSG	XRF message data set
FILEA	sample program file

*Table 6. CICS data sets created by the DFHALTDS job*

DFHAUXT	non-VSAM auxiliary trace (A) data set
DFHBUXT	non-VSAM auxiliary trace (B) data set
DFHDMPA	non-VSAM dump (A) data set
DFHDMPB	non-VSAM dump (B) data set
DFHLCD	CICS local catalog

---

## MVS system data sets used by CICS

Besides its own system data sets, summarized in Table 3 on page 24, CICS also uses some MVS data sets. These are:

Table 7. MVS data sets used by CICS

Data set	Owned or used by	Other comments
SDUMP data sets	MVS SDUMP macro	Used by CICS for system dumps through the MVS SDUMP macro.
SMF data sets	System management facility	Used by CICS monitoring and statistics domains for monitoring and statistics records.
GTF data sets	Generalized trace facility	Used by CICS trace domain for CICS trace entries.

Recalculate the size of these system data sets, taking into account the increased volumes of data that CICS generates. For example, for an SDUMP data set you need at least 25 cylinders of a 3380 device, or the equivalent. For guidance information about calculating the size of SDUMP data sets, see the *OS/390 MVS Initialization and Tuning Guide* manual.

The SDUMP data sets can become full of unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after message DFHAP0001). To prevent this, suppress such SDUMPs as described on page 100.

If you are collecting CICS interval statistics frequently, or the volume of statistics at each interval is high, then you must take this into account when sizing your SMF data sets. Similarly, you must consider the amount of CICS monitoring data that is being written when CICS monitoring classes are active.

CICS can write records to SMF of up to 32756 bytes, resulting in SMF writing spanned records to the SMF data sets. For more efficient use of DASD, you should consider creating the SMF data sets to be used by CICS with a control interval size of either 16384 bytes (16KB) or 8192 bytes (8KB). If you use other control interval sizes you must consider the trade-off between efficient use of DASD, SMF data set I/O performance and the possibility of data being lost due to insufficient SMF buffers.

If you are running CICS with GTF trace on, make allowance for CICS trace entries in the GTF data sets.

For background information about SMF, and about other SMF data set considerations, see the *OS/390 MVS System Management Facilities (SMF)*.

For programming information about CICS monitoring records and their sizes, see the *CICS Customization Guide*. For programming information about CICS statistics records and their sizes, see the *CICS Performance Guide*. For background information about GTF, see the *OS/390 MVS Diagnosis: Tools and Service Aids* manual.

---

## Setting up data sets for XRF

There are some factors that you must consider regarding both the CICS system data sets and the user application data sets when you are running CICS with XRF. These considerations are about the type of data set sharing that takes place between the active and the alternate CICS regions, and about data set allocation and disposition.

Even if you intend to run CICS with XRF=NO to begin with, you are advised to think about data set dispositions with XRF from the start.

Consider the following general points when running CICS in an XRF environment:

- An alternate CICS region can be started before an active CICS region terminates.
- The active and alternate CICS regions may be executing in different MVS images.

It follows that the status and location of the data sets used by CICS become very important. In particular, consider the following points:

- For a given **file name**, do the active and alternate CICS regions:
  - Refer to separate data sets?
  - Refer to the same data set?
- For a given data set, is it required by the alternate CICS region:
  - Before takeover occurs?
  - After takeover occurs?
- For a given data set, is it allocated:
  - At job step initiation?
  - Dynamically?
- What facilities of MVS global resource serialization (GRS) or JES3 are being used?

The allocation of data sets, and how you specify the DISP parameter, are important factors when running CICS with XRF. The point at which data sets are allocated, and whether they are shared between active and alternate CICS regions must be considered. A shared data set, in XRF terms, means one that is required by both the active and alternate CICS regions, though not necessarily concurrently. (The DD statements refer to the same data set.) In an XRF environment, CICS classifies data sets as follows:

- Actively shared
- Passively shared
- Unique

### Actively shared data sets

Actively shared data sets are required for use in both the active and alternate CICS regions. There are only two CICS system data sets in this category, called the CICS availability manager (CAVM) data sets. The active and the alternate CICS regions each open these data sets during CICS initialization, and the data sets are shared while both CICS regions are running. They are:

- CAVM control data set
- CAVM message data set

It is not usual for user application data sets to be actively shared.

## Passively shared data sets

These data sets are required by both the active and alternate CICS regions, but not at the same time. Initially, they are opened by the active CICS region, and are only opened by the alternate CICS region during or following takeover. Thus in an XRF environment, passively shared data sets are said to be “owned” by the active CICS region. The CICS system data sets in this category are the:

- CICS system definition data set (DFHCSD)
- Global catalog data set (DFHGCD)
- Temporary storage data set (DFHTEMP)
- Transient data intrapartition data set (DFHINTRA)

User data sets managed by CICS file control, and DL/I data sets, are also passively shared.

## Unique data sets

These data sets are unique to either the active **or** the alternate CICS region, and are not shared in any way. The CICS system data sets in this category are:

- CICS local catalog data set (DFHLCD)
- Dump data sets (DFHDMPx)
- Auxiliary trace data sets (DFHAUXT and DFHBUXT)

User application data sets are not usually unique.

## Data set allocation

If you define data sets to CICS and MVS using DD statements, they are allocated at job step initiation. This means that the value coded for the DISP parameter is critical for all shared data sets. However, if you are using dynamic allocation, the alternate CICS region does not allocate any data sets dynamically before takeover occurs.

### **DISP=SHR**

allows data sets to be allocated concurrently by the active and alternate CICS regions. Unfortunately, it also allows the data sets to be allocated by jobs other than the active and alternate CICS regions.

If this risk proves unacceptable for BDAM and VSAM user files and for DL/I databases, then consider using dynamic allocation with DISP=OLD.

Alternatively, this exposure can be reduced by using RACF protection, which can be applied to both user and system data sets.

### **DISP=NEW|OLD|MOD**

requests exclusive use of a data set, so it follows that it may not be possible to start the alternate CICS region before the active CICS region terminates.

**Note:** MVS does not prevent conflicting concurrent use of a data set residing on shared DASD by two or more jobs running in different MVS images, even when DISP=OLD is specified. To prevent concurrent use, you can use either global resource serialization (GRS) or JES3, to provide global data set enqueueing in a multi-MVS environment. However, when you run CICS with XRF, CICS always ensures (except for the CSD) that there is no conflicting concurrent use of data sets by an active CICS region and its alternate CICS region, even though they are running in different MVS images.

For more information about sharing the CSD, see “Sharing a CSD in a multi-MVS environment (non-RLS)” on page 71.

---

## Backup while open (BWO) of VSAM files

CICS supports the **backup while open (BWO)** facility provided by DFSMSdss and DFSMSHsm. This support enables some types of VSAM data sets to be backed up by DFSMSdss while CICS is currently updating these data sets. At the same time, CICS logs forward recovery images of any changes to these data sets on a forward recovery journal. At a later date the backup of the data set can be restored using DFSMSHsm and brought to a point of consistency by applying the forward recovery logs using a forward recovery utility such as the CICS VSAM Recovery (CICSVR).

BWO is available only for data sets accessed by CICS file control, which includes the CICS system definition (CSD) data set.

VSAM data sets that are to use this facility must reside on SMS-managed DASD, and must have an ICF catalog structure. Only VSAM ESDS, RRDS (both fixed and variable), and KSDS data sets are supported.

For DFSMS 1.3, there are two ways of defining BWO:

1. By defining the cluster with parameter BWO. This parameter can take the values TYPECICS, TYPEIMS, TYPEOTHER, and NO. TYPECICS means that it is eligible for BWO in a CICS region. The other parameters are treated as not eligible. The file resource definition is ignored (even if it conflicts).
2. If the BWO parameter is not defined, it defaults to UNDEFINED. In this case CICS looks at the file resource definition.

Clusters with data sets that are to be opened in RLS mode must have BWO specified in the cluster definition.

CICS defines a data set as eligible for BWO when a file is defined using RDO. If BACKUPTYPE=DYNAMIC is specified for a VSAM file, the file is defined as eligible for BWO when the data set is opened. BACKUPTYPE=STATIC, the default, defines a file as not eligible for BWO.

If DFSMSdss is to back up a data set that is specified with BACKUPTYPE=STATIC, all CICS files currently open for update against that data set must be closed before the backup can start.

The first time a file is opened against a VSAM base cluster data set after a CICS initial or cold start, CICS checks if BWO has been specified in the ICF catalog. If it is, it updates information in the file resource definition from the ICF catalog.

If the data sets are updated in RLS mode, BWO is managed entirely by DFSMS. When a BWO copy is made, DFSMSdss sends a message to all CICS systems on the sysplex with open ACBs for the sphere. The CICS systems keep track of all current UOWs that have updated files for the sphere. When all of these have completed, CICS writes tie-up records and notifies DFSMSdss. The copy is complete when all CICS systems have responded.

If the data sets are updated in non-RLS mode and if the value specified by BACKUPTYPE is DYNAMIC, CICS issues a call to DFSMSdfp 3.2 callable services to update the ICF catalog to indicate that the base cluster data set is eligible for BWO while it is under the control of CICS.

Any subsequent file opened against the same cluster must have the same BACKUPTYPE attribute as that of the first file opened. If a mismatch is found, the subsequent file open fails.

CICS records the fact that a VSAM base cluster data set is eligible for BWO in its base cluster block. This is remembered when all files have closed against the VSAM base cluster and across CICS warm and emergency restarts. (It is not remembered across CICS cold or initial starts.) When CICS is terminated by a controlled normal shutdown, all CICS files are closed.

When the last file open for update (and defined as eligible for BWO) is closed against a base cluster data set, the DFSMSdftp callable services update the ICF catalog to indicate that this data set is no longer eligible for BWO. This prevents BWO during the batch window between CICS sessions.

**Note:** During the batch window between CICS sessions it is possible to update CICS user data sets by batch jobs (although, to maintain data integrity, this should only be done after a controlled normal shutdown, and **never** after an uncontrolled or immediate shutdown).

Any BWO backup made during a batch window after an uncontrolled or immediate shutdown should be discarded if batch updates are made. This is because those updates are not logged to CICS forward recovery logs and therefore the BWO backup could not be forward recovered to a point of consistency.

For a normal CICS shutdown, CICS also needs a **quiesced data set**<sup>5</sup> backup to be made after the batch updates and before the data set is made available to a subsequent CICS session so that CICS forward recovery can start from a consistent point.

Before DFSMSdss and DFSMSHsm take a backup of any kind of a VSAM sphere, a call is made to examine the state of the ICF catalog to check if BWO is required. If so, the backup is made without attempting to obtain exclusive control and serialize updates to this data set.

When a backup copy of a data set is restored by DFHSM and DFDSS, and the backup was of a BWO type, the ICF catalog is updated to indicate that the data set needs to be forward recovered before it can be used. CICS checks this at data set open time and fails an FCT open if the catalog indicates that the data set is back-level.

The systems administrator must put appropriate procedures into place for BWO and for forward recovery, but these new procedures should be simpler than those currently in use. These procedures must include:

- Restoring the BWO backup and running the forward recovery utility to bring the data set to a point of consistency. (The restore requires that users do not have access to the file during the recovery process.)
- Restoring and forward recovery of data sets that may have been damaged while allocated to CICS. This operation may require backout of partially committed units of work, by CICS emergency restart.

---

5. **Quiesced data set:** A data set against which all update activity has been quiesced so that DFSMSdss can have exclusive control while a backup is made.

The systems administrator must decide which VSAM user data sets are eligible for BWO, subject to the restrictions detailed in “Restrictions on BWO” applicable to heavily-updated KSDS data sets.

## Effect of disabling activity keypointing

**Note:** This affects only non-RLS activities.

If activity keypointing is disabled in your CICS region (by specifying the system initialization parameter AKPFREQ=0), this has a serious effect on BWO support, because no tie-up records (TURs) are written to the forward recovery logs, and the data set recovery point is not updated. Therefore, forward recovery of a BWO backup must take place from the time that the data set was first opened for update. This requires that all forward recovery logs are kept since that time so that forward recovery can take place. If there are many inserts or records that change length, a lot of forward recovery could be required. If, however, a record is just updated and the length is unchanged, there is no CI split. For information about TURs and recovery points, see the *CICS Recovery and Restart Guide*.

## Restrictions on BWO

The following restrictions apply to VSAM KSDS data set types only.

If a VSAM control interval or control area split occurs while a BWO is in progress, the backup is unreliable and is discarded by DFHSM and DFDSS. During such a split, certain portions of the data set may be duplicated or not represented at all in the backup as DFDSS copies sequentially. MVS/DFP 3.2 indicates that a split has occurred in the ICF catalog. At the end of the backup, DFHSM and DFDSS check the ICF catalog, and if a split has occurred, or is still in progress, discard the backup. For this reason, certain heavily-updated VSAM KSDS data sets may not be eligible for BWO, or might be eligible only during periods of reduced activity (for example, overnight). For a KSDS data set to be eligible for BWO, the typical time between control interval or control area splits must be greater than the time taken for DFHSM and DFDSS to take a backup of the data set.

## XRF considerations

CICS XRF regions take keypoints more frequently than non-XRF regions. If BWO is used, extra activity occurs during keypoints, because:

- Extra “tie-up” records (TURs) are written to associate file names with their associated data set names, and
- The ICF catalog is updated to record the recovery time (the time from which the forward recovery utility must start applying log records).

These actions are performed approximately once every 30 minutes.

---

## Using storage management facilities

The following storage management facilities are needed to use BWO:

- Storage management subsystem (SMS), part of MVS/DFP Version 3 Release 2 or later, product number 5665-XA3.
- Data facility hierarchical storage manager (DFHSM), product number 5665-329.
- Data facility data set services (DFDSS), product number 5665-327.

For more information about these facilities see the following sections.

## Storage management subsystem (SMS)

SMS is the approach to DASD storage management in which:

- CICS, by means of the storage management subsystem, determines data placement
- An automatic data manager handles data backup, movement, space, and security

This is sometimes referred to as DFSMS and complements functions of MVS/DFP and other individual products of the Data Facility product family. For more details about SMS, see the following publications:

- *MVS/DFP Version 3 Release 2: Storage Administration Reference*, SC26-4566  
This describes storage administrator applications.
- *MVS/DFP Version 3 Release 2: General Information*, SC26-4552  
This gives an overview of MVS/DFP and its requirements and describes concepts of SMS-managed storage.

### Message on recall of backed up data sets

If you use DFHSMS to manage your VSAM data sets, you should consider carefully the period after which your CICS VSAM data sets are migrated to primary or secondary storage. If a migrated data set has to be recalled for CICS, it can take several minutes from primary storage, or longer from secondary storage. While the recall is taking place, the user is locked out, and no other opens or closes on that data set can be performed until the data set has been recalled.

If a migrated data set has to be recalled, CICS issues message DFHFC0989 to the system console, to notify the user that a recall is taking place, and to indicate whether it is from primary or secondary storage.

## Data facility hierarchical storage manager (DFHSM)

DFHSM is an IBM-licensed program to manage volumes and data sets. For more details about DFHSM, see the:

*Data Facility Hierarchical Storage Manager Version 2 Release 5.0: General Information*, GH35-0092

This discusses the main features, options, and potential benefits of DFHSM, and addresses people who want to know what the DFHSM program can do for them.

### Data facility data set services (DFDSS)

DFDSS is an IBM-licensed program used to copy, move, dump, and restore data sets and volumes. For more details about DFDSS, see the:

*Data Facility Data Set Services Version 2 Release 5.0: General Information*, GC26-4123

This introduces DFDSS and helps you evaluate its use.

---

## Chapter 4. Setting up the temporary storage data set

This chapter describes how to define the temporary storage data set. CICS provides the **temporary storage** facility to enable application programs to hold data, created by one transaction, for use later by the same transaction or by a different transaction. You save data in temporary storage queues that are identified by symbolic names. Temporary storage queues can be in main storage, VSAM-managed auxiliary storage, or temporary storage pools in the coupling facility.

You would use main storage if:

- Data is needed for only short periods of time
- Data does not need to be recoverable
- Only small amounts of data are to be stored

You would use auxiliary temporary storage if:

- Large amounts of data are to be stored
- Data is to be kept for extended periods of time
- Data is to be maintained from one CICS run to the next

You define auxiliary temporary storage as a nonindexed VSAM data set. CICS uses control interval processing when storing or retrieving temporary storage records in this data set. A control interval usually contains several records. Temporary storage space within a control interval is reusable.

Temporary storage queues can also reside in queue pools in a coupling facility. This applies to non-recoverable queues which may be written to and read from different CICS regions. For more information about temporary storage data sharing, see “Defining temporary storage pools for temporary storage data sharing” on page 38. For background information about CICS temporary storage, see the *CICS Application Programming Guide*.

---

### Defining the temporary storage data set

To define a VSAM data set for auxiliary temporary storage, as a single extent data set on a single volume, you can use the sample job shown in Figure 3 on page 36.

Alternatively, you can run the CICS-supplied job DFHDEFDS (in CICSTS31.XDFHINST), to create the DFHTEMP data set as one of the data sets for a CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*.

**Notes:**

1. You must not define any extra associations for a temporary storage data set. (Do not, for example, define a PATH.) Doing so causes CICS startup to fail.

```

//DEFTS   JOB accounting info,name
//AUXTEMP EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN   DD *
        DEFINE CLUSTER(NAME(CICSTS31.CICS.CNTL.CICSqualifier.DFHTEMP)-
        RECORDSIZE(4089,4089)           - 1
        RECORDS(200)                   -
        NONINDEXED                      -
        CONTROLINTERVALSIZE(4096)      -
        SHAREOPTIONS(2 3)               -
        VOLUMES(vol1id)                 -
        DATA(NAME(CICSTS31.CICS.CNTL.CICSqualifier.DFHTEMP.DATA) -
        UNIQUE)
/*

```

Figure 3. Sample job defining an auxiliary temporary storage data set

2. **1** The RECORDSIZE value must be 7 bytes less than the CONTROLINTERVALSIZE. See “Space considerations” for information about calculating the control interval size.
3. Do not allocate the DFHTEMP data set from an SMS data class using extended addressability. CICS does not support this.

## Using multiple extents and multiple volumes

The job control statements in Figure 3 are for a single-extent data set defined on a single volume. That data set must be big enough to hold all your data. Instead of defining one data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define multiple extents and multiple volumes. For more information about defining these, see “Multiple extents and multiple volumes” on page 26.

## Space considerations

The amount of space allocated to temporary storage is expressed in two values that you must specify:

1. The control interval size
2. The number of control intervals in the data set

### The control interval size

You specify the control interval size with the CONTROLINTERVALSIZE parameter in the VSAM CLUSTER definition. Because a control interval contains one or more temporary storage records, take the temporary storage record size into account when choosing the control interval size. The following factors affect your choice:

- Each temporary storage record **must** have space for:
  - The data
  - 36 bytes (for the temporary storage header)

If you install BMS with 3270 support, the data length of the record is at least as large as the 3270 buffer size. For 3270 terminals with the alternate screen size facility, the data length is the larger of the two sizes.

The total number of bytes allocated for a temporary storage record (including the 36-byte header) is rounded up to a multiple of 64 (for control interval sizes less than, or equal to, 16 384), or a multiple of 128 (for larger control interval sizes).

- The control interval size should be large enough to hold at least one (rounded up) temporary storage record, including 64 bytes of VSAM control information for control interval sizes less than, or equal to, 16 384, or 128 bytes of control information for larger control interval sizes. The maximum control interval size is 32 KB.

Choose a control interval size large enough to hold the largest normally occurring temporary storage record, together with the VSAM control information. Oversize records are split across control intervals, but this might degrade performance. Control interval sizes must be multiples of 512 bytes when smaller than 8 KB and must be a multiple of 2 KB when equal to or larger than 8 KB.

**Example:** If you use BMS to write a 24 x 80 character screen to temporary storage, the data written occupies 1920 bytes. You need 36 bytes for the CICS temporary storage header, giving a total of 1956 bytes. Rounding this up to a multiple of 64 gives 1984 bytes. Finally, adding a further 64 bytes of VSAM control information gives a control interval size of 2048 bytes. Typically, the CI size is larger than this, to hold several records possibly differing in size.

### Number of control intervals

VSAM uses the RECORDS and RECORDSIZE operands to allocate enough space for the data set to hold the number of records of the specified size. You must code the same value for the two operands of the RECORDSIZE parameter (the average and maximum record sizes), and this value must be 7 bytes less than the CONTROLINTERVALSIZE. In this way, the specified number of VSAM records matches the number of control intervals available to temporary storage management. You thus specify, indirectly, the number of control intervals in the temporary storage data set. (Note that the RECORDS and RECORDSIZE parameters do not correspond to the temporary storage records as seen at the CICS temporary storage interface.)

The number of control intervals to be allocated depends on user and system requirements for temporary storage, up to the maximum number permitted of 65 535.

## Number of VSAM buffers and strings

You can use the TS system initialization parameter to specify the number of CICS temporary storage buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in “Performance considerations of TS and TD buffers” on page 26. You should specify a value to suit your CICS region. If you specify **TS=(,0)**, requests for auxiliary temporary storage are executed using main storage.

## Job control statements for CICS execution

The DD name required by the temporary storage data set is DFHTEMP. For a CICS execution, you need a data definition statement for DFHTEMP in the startup job stream, such as:

```
//DFHTEMP DD DSN=CICSTS31.CICS.app1id.DFHTEMP,DISP=SHR
```

## XRF considerations

The temporary storage data set is a passively shared data set, owned by the active CICS region, but allocated to both the active and alternate CICS regions.

Although the alternate CICS region does not open this data set before takeover, it is allocated at job step initiation, so you must specify DISP=SHR on the DD statement to enable the alternate CICS region to start.

---

## Defining temporary storage pools for temporary storage data sharing

Using TS data sharing means replacing main or auxiliary storage for your TS queues with one or more TS pools, where the scope and function of each TS pool is similar to a QOR. Each TS pool is defined, using MVS cross-system extended services (XES), as a keyed list structure in a coupling facility. This means you must define the pool using the coupling facility resource manager (CFRM) policy statements. Using the CFRM policy definition utility, IXCMIAPU, you specify the size of the list structures required, and their placement within a coupling facility. For an example of this utility, see member IXCCFRMP in the SYS1.SAMPLIB library, in the *OS/390 MVS Setting Up a Sysplex* manual. An example of a definition statement is shown in Figure 4.

```
STRUCTURE NAME(DFHXQLS_PRODTSQ1)
  SIZE(1000)
  INITSIZE(500)
  PREFLIST(FACIL01,FACIL02)
```

Figure 4. Example of defining the estimated size of a list structure

The name of the list structure for a TS data sharing pool is created by appending the TS pool name to the prefix DFHXQLS\_, giving DFHXQLS\_*poolname*. When defined, you must activate the CFRM policy using the MVS operator command SETXCF START.

When a list structure is allocated, it may have an initial size and a maximum size specified in the CFRM policy. (All structure sizes are rounded up to the next multiple of 256K at allocation time). Provided that space is available in the coupling facility, a list structure can be dynamically expanded from its initial size towards its maximum size, or contracted to free up coupling facility space for other purposes. Note that if the initial structure allocation becomes full, the structure does not expand automatically, even if the structure allocated is less than the specified maximum size. To expand a list structure when the allocation becomes full, you can expand it (up to its maximum size) using the following SETXCF command:

```
SETXCF START,ALTER,STRNAME=DFHXQLS_poolname,SIZE=nnnn
```

## Approximate storage calculations

```
# A coupling facility structure contains not only stored data but also the information
# needed to manage and access that data, in a similar way to a key-sequenced data
# set. The data for each entry in the coupling facility is stored as a chain of fixed size
# (usually 256-byte) elements, which means that the exact length for variable-length
# data has to be stored separately. CICS does this by including a length prefix in the
# stored data, so space calculations have to allow for each entry using a whole
# number of elements. The amount of internal control information depends on the
# level of functionality and performance of the coupling facility control code for the
# current CFLEVEL. The storage requirements can increase for a higher CFLEVEL.
# For more information about how the coupling facility structure storage is organized,
# see
```

```
# The easiest way to calculate accurate structure storage requirements is to use the
# web-based IBM CFSizer tool at http://www-1.ibm.com/servers/eserver/zseries/
# cfsizer/. The tool takes these factors into account and communicates with a
# coupling facility at a current CFLEVEL to perform the calculation. You need to enter
# some minimum input to get an accurate storage calculation as follows:
```

# **MAXQUEUES**  
# specifies the maximum number of data lists that are reserved when the  
# structure list is allocated by CICS. This determines the maximum number of  
# large queues that can be stored in the structure and corresponds to the  
# MAXQUEUES server parameter. A large queue is where the total size of  
# the data items exceeds 32K. It is stored in a separate list in the structure.  
# Although you should specify a large enough number, specifying an  
# excessively large number will use up an unnecessary amount of coupling  
# facility storage for unused preallocated list headers. The valid range is from  
# 1 to 999999. The default is 1000.

# **Average rounded item size**  
# specifies the average amount of storage required for each TS queue item.  
# Note that each item has a two-byte length prefix and is stored as one or  
# more 256-byte elements. This number should range from 1 to 32768. This  
# value determines the entry to element ratio that is used to calculate the  
# required structure size.

# If all queue items are approximately the same size, calculate this value by  
# taking the average data size, adding two, and rounding up to the next  
# multiple of 256. If queue items are different sizes, round up each size first  
# before taking the average. For example, if half of the items are 100 bytes  
# and half are 300 bytes, then the rounded sizes are 256 and 512. The  
# average rounded item size is half way between those values, 384, which is  
# more accurate than taking the overall average item size of 200 and  
# rounding it up to 256.

# **Total number of items in all queues**  
# specifies the total number of entries in all of the TS queues.

# **Target Usage Percent**  
# specifies the percentage of the structure space which the given total  
# number of items are expected to use. Specify a number in the range of 1 to  
# 100. The default is 75. This leaves some free space for temporary  
# expansion, and to give time to expand the structure in response to warning  
# messages (which normally start at 80%) if the initial free space is not  
# enough.

# **Max Expansion Percent**  
# specifies the percentage that the structure can expand. If a non-zero value  
# is specified here, the maximum structure size will be greater than the initial  
# structure size by an amount sufficient to allow for a further expansion of the  
# total amount of data by this percentage. For example, if the value 200 is  
# specified, the initial size will be enough to store the specified total number  
# of items, but the maximum size will be enough to store three times that  
# number of items.

Alternatively you can use the following calculation, which applies up to CFLEVEL 11 of the coupling facility. If you have a coupling facility at CFLEVEL 12, you need to change the calculation to add 10% and 2MB to the sizings in the formula below.

### Storage calculations

$$\text{Item entry size} = (170 + (\text{average item size, rounded up to next 256})) \\ + 5\% \text{ extra for control information}$$
$$\text{Total size} = 200\text{K} \\ + (\text{number of large queues} \times 1\text{K}) \\ + (\text{number of items in all queues} \times \text{item entry size})$$

The calculation determines the size of a structure that would be approximately 100% full for the specified level of usage. For practical operation, however, a reasonable proportion of free space must be available, not only to minimize the risk of the structure becoming totally full but also to avoid triggering low space warning messages and additional activity to alter entry to element ratios. The maximum normal usage should therefore aim to be about 75% of the structure size. Expected usage values should therefore be adjusted upwards to allow for the required amount of free space (for example, by about one third to aim for 75% usage).

#### Notes:

1. If you have a small queue with multiple items that does not exceed 32K, then all the queue items are stored as the data portion of the queue index entry. If the queue exceeds 32K, then it is converted to a form where one item per entry is stored in a separate list and referred to by the queue index entry.
2. The amount of element storage required is two bytes more than the data item size due to the length prefix on each item.
3. Defining the CFRM policy statements for a list structure does not actually create the list structure, this is done by a TS server during its initialization.

For information about the TS server system initialization parameters, see Chapter 24, "Setting up and running a temporary storage server," on page 369.

For information about defining list structures, see the following MVS publications:  
*OS/390 MVS Setting Up a Sysplex*, GC28-1779  
*OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771  
*OS/390 MVS Programming: Sysplex Services Reference*, GC28-1772

---

## Chapter 5. Setting up transient data destination data sets

Data sets used for transient data destinations (queues) can be intrapartition or extrapartition. This chapter tells you how to define data sets for transient data queues. The transient data intrapartition data set is a VSAM entry-sequenced data set (ESDS) used for queuing messages and data within the CICS region. Transient data extrapartition data sets are sequential files, normally on disk or tape; each queue can be used either to send data outside the CICS region or to receive data from outside the region. For background information about intrapartition and extrapartition transient data, see the *CICS Application Programming Guide*.

Messages or other data are addressed to a symbolic queue which you define as either intrapartition or extrapartition using the CEDA transaction. The queues can be used as **indirect** destinations to route messages or data to other queues.

For information about coding transient data resources, see the *CICS Resource Definition Guide*.

System messages that CICS produces are commonly sent to transient data queues, either intrapartition or extrapartition. The following is a brief description of the queues used by CICS. Sample definitions for the CICS-supplied queues can be found in group DFHDCTG, which is included in list DFHLIST.

<b>CADL</b>	CEDA VTAM resource log, indirect to CSSL.
<b>CADO</b>	AD resource log, indirect to CSSL.
<b>CAIL</b>	Autoinstall terminal model resource log, indirect to CSSL.
<b>CCPI</b>	CPI Communications message log, indirect to CSSL.
<b>CCSE</b>	C error data stream (stderr) log, indirect to CCSO.
<b>CCSO</b>	C output data stream (stdout) log, direct to COUT.
<b>CCZM</b>	Messages for CICS classes, indirect to CSSL.
<b>CDB2</b>	CICS DB2 message log, indirect to CSSL.
<b>CDBC</b>	Database log, indirect to CSSL.
<b>CDUL</b>	Dump message log, indirect to CSSL.
<b>CEJL</b>	Enterprise bean message log, indirect to CSSL.
<b>CESE</b>	Language Environment error data stream (stderr), direct to CEEMSG.
<b>CESO</b>	Language Environment output data stream (stdout), direct to DD name CEEOUT.
<b>CIEO</b>	ECI over TCP/IP message log, indirect to CSSL.
<b>CIIL</b>	IIOF message log, indirect to CSSL.
<b>CJRM</b>	Java Remote Access Services message log, indirect to CSSL.
<b>CMIG</b>	Migration log to detect use of EXEC CICS ADDRESS CSA commands, indirect to CSSL.
<b>CPLD</b>	PL/I dumps, indirect to CPLI.
<b>CPLI</b>	PL/I SYSPRINT output, direct to DDname PLIMSG.
<b>CRDI</b>	RDO install log, indirect to CSSL.
<b>CRPO</b>	ONC/RPC message log, direct to DDname CRPO.
<b>CSBA</b>	CICS BTS message log, indirect to CSSL.
<b>CSBR</b>	Bridge message log, indirect to CSSL.
<b>CSCC</b>	CICS Client message log, indirect to CSSL.
<b>CSCS</b>	Signon/sign-off security log, indirect to CSSL.
<b>CSDH</b>	Document Domain message log, indirect to CSSL.
<b>CSDL</b>	CEDA command log, indirect to CSSL.
<b>CSFL</b>	File allocation message log, indirect to CSSL.
<b>CSJE</b>	JVM error data stream (stderr), indirect to CSSL.
<b>CSJO</b>	JVM output data stream (stdout), indirect to CSSL.

<b>CSKL</b>	Transaction and profile resource log, indirect to CSSL.
<b>CSML</b>	Signon/sign-off message log, indirect to CSSL.
<b>CSMT</b>	Terminal error message and transaction abend message log, indirect to CSSL.
<b>CSNE</b>	ZNAC-produced messages log, indirect to CSSL.
<b>CSOO</b>	Sockets message log, indirect to CSSL.
<b>CSPL</b>	Program resource log, indirect to CSSL.
<b>CSQL</b>	Transient data message log, indirect to CSSL.
<b>CSRL</b>	Partner resource log, indirect to CSSL.
<b>CSSH</b>	Scheduler message log, indirect to CSSL.
<b>CSSL</b>	Message log, direct to DD name MSGUSR (all the other general CICS queues are defined as indirect queues to CSSL).
<b>CSTL</b>	Terminal I/O error log, indirect to CSSL.
<b>CSZL</b>	The queue used for Front End Programming Interface (FEPI) messages, indirect to CSSL. You do not have to define this queue if you do not have FEPI installed, but it is included in DFHDCTG.
<b>CSZX</b>	The queue used for Front End Programming Interface (FEPI) processing, indirect to CSSL. You do not have to define this queue if you do not have FEPI installed, but it is included in DFHDCTG.
<b>CWBO</b>	CICS Web support messages, indirect to CSSL.
<b>CWBW</b>	HTTP warning headers on messages received by CICS Web support, indirect to CSSL.

You should include in your CICS region all the queues that CICS uses. Although the omission of any of the queues does not cause a CICS failure, you lose important information about your CICS region if CICS cannot write its data to the required queue. Sample definitions of all the queues that CICS uses can be found in group DFHDCTG, which is included in list DFHLIST and is unlocked so that you can alter the definitions.

**Note:** We recommend that you take a backup copy of the changes made to DFHDCTG in case maintenance is applied.

For information about the queues used by CICS, see the *CICS Resource Definition Guide*.

For information about the queues that CICS uses for RDO, see “Multiple extents and multiple volumes” on page 26.

For a way of printing these system messages on a local printer as they occur, see the transient data write-to-terminal sample program, DFH\$TDWT. This sample program is supplied with the CICS pregenerated system in CICSTS31.CICS.SDFHLOAD, and the assembler source is in CICSTS31.CICS.SDFHSAMP. For programming information about DFH\$TDWT, see the *CICS Customization Guide*.

---

## Defining intrapartition data sets

You use job control statements to define the transient data intrapartition data set, which must be big enough to hold all the data for intrapartition queues. You can define the data set as any one of the following:

- One extent on one volume
- Multiple extents on one volume
- One extent on each of several volumes
- Multiple extents on multiple volumes

Figure 5 shows job control statements to define a single extent data set on a single volume. Instead of defining one extent data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define multiple extents and/or multiple volumes. For considerations about using multiple extents and/or multiple volumes, see “Using multiple extents and multiple volumes.”

```

//DEFDS      JOB  accounting info,name,MSGCLASS=A
//TDINTRA    EXEC PGM=IDCAMS
//SYSPRINT   DD   SYSOUT=A
//SYSIN      DD   *
              DEFINE CLUSTER -
                  ( NAME(CICSTS31.CICS.applid.DFHINTRA)  -
                    RECORDSIZE(1529,1529)              -
                    RECORDS(100)                       -
                    NONINDEXED                         -
                    CONTROLINTERVALSIZE(1536)          -
                    VOL(volid)                        -
                  DATA -
                    ( NAME(CICSTS31.CICS.applid.DATA.DFHINTRA))
/*
//

```

Figure 5. Sample job to define a transient data intrapartition data set

## Job control statements to define the intrapartition data set

For an example of the JCL that you can use to define the transient data intrapartition data set, see Figure 5. If you allocate space in records, rather than tracks or cylinders, you need RECORDSIZE, and it should be 7 bytes less than the CONTROLINTERVALSIZE. (For full details, see the appropriate VSAM manuals.)

Alternatively, you can run the CICS-supplied job DFHDEFDS to create the DFHINTRA data set as one of the data sets for a CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*.

**Note:** You must not define any extra associations for a transient data intrapartition data set. (Do not, for example, define a PATH.) Doing so causes CICS startup to fail.

### What happens if the intrapartition data set fails to open

The DFHINTRA data set is opened during CICS initialization, regardless of the presence (or absence) of any intrapartition queue definitions that might become active during GRPLIST installation. If DFHINTRA fails to open during an initial or cold start of CICS, a message is issued informing you of this, and asking you if you wish to continue or if you wish to cancel CICS.

If DFHINTRA opened successfully during a previous startup but fails to open during a subsequent warm or emergency restart, CICS is terminated.

If CICS initializes without a DFHINTRA data set, any attempts to install intrapartition data destinations for that run of CICS fails and appropriate error messages are issued.

### Using multiple extents and multiple volumes

The job control statements in Figure 5 are for a single extent data set defined on a single volume. That data set must be big enough to hold all your data. Instead of

defining one data set, which might have to be much larger than your average needs to cater for exceptional cases, you can define multiple extents and multiple volumes. For more information about defining these, see “Multiple extents and multiple volumes” on page 26.

## Space considerations

Space is allocated to queues in units of a control interval. The first CI is reserved for CICS use, the remaining CIs are available to hold data. Data records are stored in CIs according to VSAM standards.

The CONTROLINTERVALSIZE parameter must be large enough to hold the longest record, plus the 32 bytes that CICS requires for its own purposes. The maximum control interval size is 32KB.

### Size of the intrapartition data set

- If all available control intervals are currently allocated to queues, further EXEC CICS WRITEQ TD requests receive a NOSPACE response until control intervals are released by READQ TD or DELETEQ TD requests.
- The intrapartition data set should hold at least two control intervals.
- When a logically recoverable queue is read until a QZERO condition is returned, and the request is committed, CICS retains the last CI used by the queue (unless there is no further space between the end of the last record and the end of the CI). Retaining the final CI used by the queue, means that subsequent requests to write to the queue can be accommodated in the remaining space within the CI if they can fit there. This avoids the need to acquire a new CI whenever the first request is made to write to the queue following a QZERO condition, which benefits performance on subsequent write requests. However, the CI is left allocated to the queue, and so increases the usage of the data set, and the possibility of a NOSPACE condition being returned by CICS.

### Intrapartition data set restriction

A transient data intrapartition data set should be associated with one, and only one, CICS region. (If you are running CICS with XRF, one region means a pair of active and alternate CICS regions.) The destination control table contains relative byte addresses (RBAs) of records written to an intrapartition data set, and care must be taken to preserve the RBAs during any VSAM export or import operation on the data set.

Data can be corrupted or lost if:

- You start CICS with the wrong intrapartition data set; that is, a data set that contains data from another CICS region.
- You use VSAM export or import services to:
  - Increase the available space by compressing the data set, or
  - Increase the control interval size

## Number of VSAM buffers and strings

You can use the TD system initialization parameter to specify the number of CICS transient data buffers up to the maximum of 32 767. The number of buffers that you specify may have an effect on CICS performance, as described in “Performance considerations of TS and TD buffers” on page 26. You should specify a value to suit your CICS region.

## Job control statements for CICS execution

The DD name for the intrapartition data set is DFHINTRA, and the DSN operand must be the name of the VSAM entry-sequenced data set. For a CICS execution, you need a data definition statement such as:

```
//DFHINTRA DD DSN=CICSTS31.CICS.app1id.DFHINTRA,DISP={OLD|SHR}
```

## XRF considerations

A transient data intrapartition data set is a passively shared data set, owned by the active CICS region, but allocated to both active and alternate CICS regions.

Although the alternate CICS region does not open this data set before takeover, it is allocated at job step initiation, therefore specify DISP=SHR on the DD statement.

---

## Defining extrapartition data sets

#  
#  
#  
#

You can define each extrapartition data set either for input only or for output only, but not for both. You should define transient data extrapartition data sets used as queues for CICS messages with a record length of 132 bytes and a record format of V or VB. If you use the FREE=CLOSE parameter for an input extrapartition entry, be aware that it will only be usable once in the CICS session. Any attempt to read the queue after it has been CLOSEd and OPENed again will result in the IOERR condition being raised.

```
//LOGUSR DD DSN=CICSTS31.CICS.applid.LOGUSR,DISP=(NEW,KEEP),  
//      DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),  
//      VOL=SER=valid,UNIT=3380,SPACE=(CYL,2)  
//MSGUSR DD SYSOUT=A  
//PLIMSG DD SYSOUT=A
```

Figure 6. Sample JCL to define transient data extrapartition data sets

## The DFHCXRF data set

Besides any extrapartition data sets that you might define, there is a special extrapartition queue that CICS creates dynamically. This has the identifier CXRF, and is created by CICS early in the initialization process. The DD name for this extrapartition data set is DFHCXRF. You cannot define CXRF or DFHCXRF. If you code DFHCXRF as a DSCNAME, or code CXRF as a destination identifier, an error message is issued. If you create a definition for CXRF in the CSD, CICS does not install the definition. This is because the CICS entry is hardcoded and cannot be removed or replaced.

Although the CXRF data set has special significance in an alternate CICS region when you are operating CICS with XRF, it is also available in an active CICS region, and CICS regions running with XRF=NO.

If an attempt is made to write to a CICS-defined transient data queue before CICS is fully initialized, a message is written to CXRF.

If, on an initial or cold start, a request is received to write a record to a queue that has not yet been installed (as part of GRPLIST), the record is written to CXRF.

If an attempt is made to write to an intrapartition queue after the warm keypoint has been taken, the record is written to CXRF.

### DFHCXRF data set in active CICS regions

CICS uses the CXRF queue during CICS initialization as some CICS components may need to write to transient data queues. If the queues are not available at initialization time, the request to write to these queues is redirected to CXRF. Any requests from CICS components to write to transient data before the CXRF queue definition has been installed fails with a QIDERR condition.

Any requests to write to an intrapartition transient data queue after the warm keypoint has been taken during a normal shutdown are routed to CXRF.

If you want to take advantage of the special CXRF queue, you must include a DD statement for DFHCXRF. (For example, see Figure 7 on page 47.) If you omit the DD statement, transient data write requests redirected to CXRF fail with a NOTOPEN condition.

## DFHCXRF data set in alternate CICS regions

DFHCXRF has special significance for alternate CICS regions, because transient data initialization is suspended while the alternate CICS region is in standby mode, and is not completed until a takeover occurs. If you want to capture messages written to transient data by the alternate CICS region before takeover, you must include a DD statement defining the DFHCXRF data set. These messages include information about terminals that have been installed and logged on to the active CICS region. The alternate CICS region takes this information from the message data set and stores it in the CICS-generated extrapartition transient data queue, CXRF.

## DD statements for the DFHCXRF data set

You can define the DFHCXRF data set to MVS in the same way as other transient data extrapartition data sets, either to a SYSOUT data set or a sequential data set on disk (or tape). For example, you could use either of the DD statements shown in Figure 7 in a startup job stream for an alternate CICS region.

```
//DFHCXRF DD SYSOUT=*  
  
or  
  
//DFHCXRF DD DSN=CICSTS31.CICS.applid.DFHCXRF,DISP=(NEW,KEEP),  
//          DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),  
//          VOL=SER=volid,UNIT=3380,SPACE=(TRK,5)
```

Figure 7. Sample DD statements for DFHCXRF

Before takeover occurs, the alternate CICS region assumes that the transient data queues are defined as indirect, and pointing to CXRF. CXRF is associated with the data set that has the DD name DFHCXRF.

## XRF considerations

Except for DFHCXRF, an alternate CICS region does not open any extrapartition data sets before takeover. (See “The DFHCXRF data set” on page 46.)

Normally, when data sets are defined for output, you should have separate data sets for the active and alternate CICS regions; that is, they are unique data sets in CICS terms.

If you have separate data sets, you can code the following DISP operands in the DD statements that define the data sets:

```
DISP=SHR  
DISP=NEW  
DISP=OLD  
DISP=MOD
```

Whatever you code on the DISP parameter, be aware that data might be lost when the alternate CICS region takes over from the active CICS region, because takeover involves abending or canceling the active CICS region.

If you do not have separate data sets, you should code DISP=SHR. Anything else implies exclusive use of the data set, and for this reason you could not start an alternate CICS region (in the same MVS image as the active CICS region) until the active CICS region terminates.

Data written by the active CICS region is lost when the alternate CICS region takes over and opens the data set.

---

## Chapter 6. Setting up CICS log streams

This chapter describes how to define and create CICS log streams that exploit the MVS system logger to record journaling and logging information. CICS log manager supports:

- The CICS system log, which is also used for dynamic transaction backout.
- User journals, forward recovery logs, and autojournals. These are general logs.

This chapter covers the following topics:

- “Defining CICS system logs”
- “Defining CICS general logs” on page 51
- “Naming journals” on page 53
- “Defining JOURNALMODELS” on page 55
- “Mapping log streams” on page 56
- “Using the Journal utility program, DFHJUP” on page 61

---

### Defining CICS system logs

Each CICS region requires its own system log. The system log is implemented as two MVS system logger log streams - a primary and a secondary - but, together, they form a single logical log stream.

The system log is used for recovery purposes - for example, during dynamic transaction backout, or during emergency restart, and is not meant to be used for any other purpose.

CICS connects to its system log automatically during initialization (unless you specify a journal model definition that defines the system log as type DUMMY).

You must define a system log if you want to preserve data integrity in the event of unit of work failures and CICS failures. CICS needs a system log in order to perform:

- The backout of recoverable resources changed by failed units of work.
- Cold starts when CICS needs to recover conversation state data with remote partners.
- Warm restarts, when CICS needs to restore the region to its pre-shutdown state.
- Emergency restarts, when CICS needs to restore the region to its pre-shutdown state as well as recovering transactions to perform the backout of recoverable resources changed by units of work that were in-flight at the time of shutdown.

For information on how to define CICS system log streams, see coupling facility log streams and DASD-only log streams in the *CICS Transaction Server for z/OS Installation Guide*.

### Planning your CICS system log streams

A CICS system log (which comprises two physical log streams) is unique to the region and must not be used by any other CICS region. The default log stream names *region\_userid.applid.DFHLOG* and *region\_userid.applid.DFHSHUNT* are designed to ensure unique names.

#### Using JOURNALMODELS to define the system log

You might want to use journal models for system logs if the CICS region userid changes between runs (for example, where a test CICS region is shared between

application developers). It would be wasteful to create log streams with a different high level qualifier for each user. Using the same system log stream regardless of the which programmer starts up the CICS region keeps the number of log streams to a minimum. The following example uses a specific JOURNALNAME, with symbols in the STREAMNAME, making this an explicit model for the primary log stream.

```
DEFINE GROUP(TEST) DESC('System logs for test CICS regions')
    JOURNALMODEL(DFHLOG) JOURNALNAME(DFHLOG) TYPE(MVS)
    STREAMNAME(TESTCICS.&APPLID..&JNAME.)
```

If you define JOURNALMODEL resource definitions to define log stream names for DFHLOG and DFHSHUNT, ensure that the resulting log stream names are unique. If you have some CICS regions that use the same applid, you must use some other qualifier in the log stream name to ensure uniqueness.

If you use JOURNALMODEL resource definitions for the system log, these resource definitions must be defined and added to the appropriate group list (using the CSD utility program, DFHCSDUP) before INITIAL-starting CICS.

System logs cannot be TYPE(SMF).

DFHLOG can be TYPE(DUMMY), but you can use this only if you always INITIAL start your CICS regions and there are no recoverable resources requiring transaction backout. CICS cannot perform a cold start, or warm or emergency restart if TYPE(DUMMY) is specified on the JOURNALMODEL definition.

If you do not want to use a system log, perhaps in a test or development region, define a JOURNALMODEL for DFHLOG with type DUMMY, as shown in the following example:

```
DEFINE JOURNALMODEL(DFHLOG) GROUP(CICSLG)
    JOURNALNAME(DFHLOG)
    TYPE(DUMMY)
```

To start a CICS region without a system log, you must ensure that a JOURNALMODEL definition, such as the one shown above, is included in the start-up group list. Use the DFHCSDUP batch utility program to define the required JOURNALMODEL and to add the group to the group list.

DFHSHUNT can be TYPE(DUMMY). This is not recommended, however, because it impairs the ability of CICS to manage the system log.

### **Effects of the AKPFREQ parameter**

Review the activity keypoint frequency (AKPFREQ) defined for each CICS region. The larger the value, the more coupling facility space you need for the system logs, but you should not set AKPFREQ too low so that transactions last longer than an activity keypoint interval.

---

## Defining CICS general logs

Journals on general log streams comprise user journals, forward recovery logs, and autojournals.

## Planning log streams for use by your user journals and autojournals

General logs are identified as such by their MVS log stream names, which are differentiated from system log streams in that their names do not end with DFHLOG or DFHSHUNT.

### Using JOURNALMODELS to define general logs

If you are running multiple cloned copies of your application-owning regions (AORs), it is probable that the logged data is common and that you would want to merge the data from all of the AORs to the same log stream. The following JOURNALMODEL resource definition maps CICS journals of the same journal identifier to a shared log stream:

```
DEFINE GROUP(MERGE) DESC('Merge journals across cloned CICS regions')
  JOURNALMODEL(JRNLS) JOURNALNAME(DFHJ*) TYPE(MVS)
  STREAMNAME(&USERID..SHARED.&JNAME.)
```

In this example, the literal SHARED is used in place of the default CICS applid, which would require a unique log stream for each region.

You might want to use JOURNALMODELS to map journals to log streams if the CICS region userid changes between runs. This could be the case, for example, where CICS test regions are shared between groups of developers. It would be wasteful to create log streams with a different high level qualifier for each user and you might prefer to use the same log streams regardless of which developer starts up a CICS region. For example, the following generic JOURNALMODEL definition maps all journals not defined by more explicit definitions to the same log stream

```
DEFINE GROUP (TEST) DESC('Journals for test CICS regions')
  JOURNALMODEL(JRNLS) JOURNALNAME(*) TYPE(MVS)
  STREAMNAME(TESTCICS.&APPLID..&JNAME.)
```

You might want to merge data written by CICS regions using different journal names to a single log stream.

```
DEFINE GROUP (TEST) DESC('Merging journals 10 to 19')
  JOURNALMODEL(J10T019) JOURNALNAME(DFHJ1*) TYPE(MVS)
  STREAMNAME(&USERID..MERGED.JNLS)
DEFINE GROUP (TEST) DESC('Merging journalnames JNLxxxxx')
  JOURNALMODEL(JNLXXXXX) JOURNALNAME(JNL*) TYPE(MVS)
  STREAMNAME(&USERID..MERGED.JNLS)
```

The last qualifier of the stream name is used as the CICS resource name for dispatcher waits. Therefore, if it is self-explanatory, it can be helpful when interpreting monitoring information and CICS trace entries.

## Planning log streams for use by your forward recovery logs

CICS performs the logging for RLS and non-RLS data sets. You can share a forward recovery log stream between multiple data sets - you do not need to define a log stream for each forward-recoverable data set. Your decision is a balance of transaction performance, rapid recovery, and the work involved in managing a large number of log streams.

For a data set open in RLS mode, the MVS logger merges all the forward recovery log records from the various CICS systems on to the shared forward recovery log.

Some points to consider are:

- All data sets used by one transaction should use the same log stream (to reduce the number of log streams written to at syncpoint).
- A good starting point is to use the same forward recovery log ID that you use in an earlier CICS release.
- Share a forward recovery log stream between data sets that:
  - Have similar security requirements
  - Have similar backup frequency
  - Are likely to need restoring in their entirety at the same time
- Log stream names should relate to the data sets. For example, PAYROLL.data\_sets could be mapped to a forward recovery log named PAYROLL.FWDRECOV.PAYLOG.
- The last qualifier of the stream name is used as the CICS resource name for dispatcher waits. Therefore, if it is self-explanatory, it can be helpful when interpreting monitoring information and CICS trace entries.
- Don't mix high update frequency data sets with low update frequency data sets, because this causes a disproportionate amount of unwanted log data to be read during recovery of low frequency data sets.
- Don't put all high update frequency data sets on a single log stream because you could exceed the throughput capacity of the stream.
- If you define too many data sets to a single log stream, you could experience frequent structure-full events when the log stream can't keep up with data flow.
- Redundant data should be deleted from log streams periodically. On OS/390 Release 2 or earlier, this is a user responsibility, and must be done before the system logger inventory entry exceeds the limit of 168 data sets per log stream. On OS/390 Release 3 or later, you can specify that redundant data is to be deleted automatically from general log streams, by defining them with AUTODELETE(YES) RETPD(dddd). For information about the AUTODELETE and RETPD MVS parameters, see the *CICS Transaction Server for z/OS Installation Guide*.

Typically, for a forward recovery log, deletion of old data is related to the data backup frequency. For example, you might keep the 4 most recent generations of backup, and when you delete a redundant backup generation you should also delete the corresponding redundant forward recovery log records. These are the records older than the redundant backup because they are no longer needed for forward recovery.

For information about CICSVR, the IBM CICS VSAM Recovery product, see the *CICS System Definition Guide*.

## Planning log streams for use by your log of logs (DFHLGLOG)

The log of logs is written by CICS to provide information to forward recovery programs such as CICS VSAM Recovery (CICSVR). The log of logs is a form of user journal containing copies of the tie-up records written to forward recovery logs. Thus it provides a summary of which recoverable VSAM data sets CICS has used, when they were used, and to which log stream the forward recovery log records were written.

If you have a forward recovery product that can utilize the log of logs, you should ensure that all CICS regions sharing the recoverable data sets write to the same log of logs stream.

```
DEFINE GROUP(JRNL) DESC('Merge log of logs')
        JOURNALMODEL(DFHLGLOG) JOURNALNAME(DFHLGLOG) TYPE(MVS)
        STREAMNAME(&USERID..CICSVR.DFHLGLOG)
```

**Note:** Note that this definition is supplied in group DFHLGMOD in DFHLIST.

If you don't have a forward recovery product that can utilize the log of logs you can use a dummy log stream:

```
DEFINE GROUP(JRNL) DESC('Dummy log of logs')
        JOURNALMODEL(DFHLGLOG) JOURNALNAME(DFHLGLOG) TYPE(DUMMY)
```

Do not share the log of logs between test and production CICS regions, because it could be misused to compromise the contents of production data sets during a restore.

---

## Naming journals

The journals have the following naming conventions.

### System logs

DFHLOG and DFHSHUNT are the journal names for the CICS system log. CICS automatically creates journal table entries for DFHLOG and DFHSHUNT during initialization as shown in Table 8.

*Table 8. Journal name entry for the CICS primary system log*

Journal table entry - CICS system log	Created during system initialization
Name: DFHLOG	Always DFHLOG for the primary log
Status: Enabled	Set when journal entry created
Type: MVS	The default, but it can be defined as DUMMY on JOURNALMODEL definition (DUMMY = no output)
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid..&applid..DFHLOG, but this can be user-defined on a JOURNALMODEL definition

### Forward recovery logs

For non-RLS data sets that have not specified their recovery attributes in the VSAM catalog, forward recovery log names are of the form DFHJnn where nn is a number in the range 1–99. You define the name of the forward recovery log in the forward recovery log id (FWDRECOVLOG) in the FILE resource definition.

User applications can use a forward recovery log through a user journal name that maps on to the same log stream name. In this case, the user records are merged on to the forward recovery log. See Table 9 for an example of this.

*Table 9. Example of journal name entry for non-RLS mode forward recovery log*

Journal table entry - forward recovery log	Entry created during file-open processing
Name: DFHJ01	Name derived from FWDRECOVLOG identifier. For example, FWDRECOVLOG(01) = DFHJ01, thus FWDRECOVLOG(nn) = DFHJnn
Status: Enabled	Set when journal entry created
Type: MVS	The default, but it can be defined as DUMMY on JOURNALMODEL definition (DUMMY = no output)

Table 9. Example of journal name entry for non-RLS mode forward recovery log (continued)

Journal table entry - forward recovery log	Entry created during file-open processing
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid.&applid.DFHJ01, but this can be user-defined on a JOURNALMODEL definition

**Note:** There is no journal table entry for the forward recovery log of an RLS file. The recovery attributes and LSN are obtained directly from the VSAM catalog, and the LSN is referenced directly by CICS file control. Therefore there is no need for indirect mapping through a journal name.

You can also choose to specify the recovery attributes and LSN for a non-RLS file in the VSAM catalog.

## User journals

CICS user journals are identified by their journal names (or number, in the case of DFHJnn names), which map on to MVS log streams.

You name your user journals using any 1-8 characters that conform to the rules of a data set qualifier name. Apart from the user journal names that begin with the letters DFHJ, followed by two numeric characters, you should avoid using names that begin with DFH. User journal names of the form DFHJnn are supported for compatibility with earlier CICS releases.

Although 8-character journal names offer considerable flexibility compared with the DFHJnn name format of previous releases, you are recommended not to create large numbers of journals (for example, by using the terminal name or userid as part of a program-generated name).

Journal name DFHLOG (on an EXEC CICS WRITE JOURNALNAME command) indicates that you want to write to the CICS system log.

When used in FILE and PROFILE resource definitions, the journal numbers 1 through 99 map on to the user journal names DFHJ01–99. You can map these journal names to specific MVS log stream names by specifying JOURNALMODEL resource definitions, or allow them to default. If you do not specify matching JOURNALMODEL definitions, by default user journals are mapped to LSNs of the form *userid.applid.DFHJnn*.

Table 10 shows an example of a user journal name table entry.

Table 10. Example of a user journal name entry for output to MVS

Journal table entry - user journals	Entry created on first reference
Name: JRNL001	Name derived from API WRITE JOURNALNAME command
Status: Enabled	Set when journal entry created
Type: MVS	This journal is defined for MVS output by a JOURNALMODEL that references the JRNL001 name
LSN: log_stream_name	By default, log_stream_name resolves to &reguserid.&applid.JRNL001, but this can be user-defined on a JOURNALMODEL definition

## Installing system log and journal names

The journal control table of earlier releases is obsolete, and is replaced by a journal names table that CICS creates dynamically.

The CICS log manager needs the name of the log stream that corresponds to a CICS system log or general log, and the type - whether it is MVS, SMF, or a dummy. Except for VSAM forward recovery log stream names taken directly from the ICF catalog, CICS maintains this information in the journal names table, together with the corresponding log stream token that is returned by the MVS system logger when CICS successfully connects to the log stream.

---

## Defining JOURNALMODELS

CICS uses JOURNALMODEL definitions to resolve log stream names at the following times:

### System log

During initialization, on an initial start only.

On a cold, warm or emergency restart, CICS retrieves the log stream name from the CICS global catalog.

### General logs (excluding log streams defined in the ICF catalog)

When a journal name is first referenced after the start of CICS, or when it is first referenced again after its log stream has been disconnected. Log stream disconnection, requiring further reference to a matching JOURNALMODEL resource definition, occurs as follows:

#### User journals

As soon as you issue a DISCARD JOURNALNAME command.

Any further references to the discarded journal name means that CICS must again resolve the log stream name by looking for a matching JOURNALMODEL resource definition. You can change the log stream name for a user journal by installing a modified JOURNALMODEL definition.

#### Auto journals for files

All files that are using the log stream for autojournaling are closed.

#### Forward recovery logs (excluding those defined in the ICF catalog)

All files that are using the log stream for forward recovery logging are closed.

A JOURNALMODEL definition generally specifies a generic journal name, thereby mapping to the same MVS log stream any journal names that match on the generic name. JOURNALMODEL definitions can also be specific models and, using JOURNALMODEL definitions, you can map many journals or forward recovery logs to the same MVS log stream, or assign them to SMF (see Figure 8 on page 56).

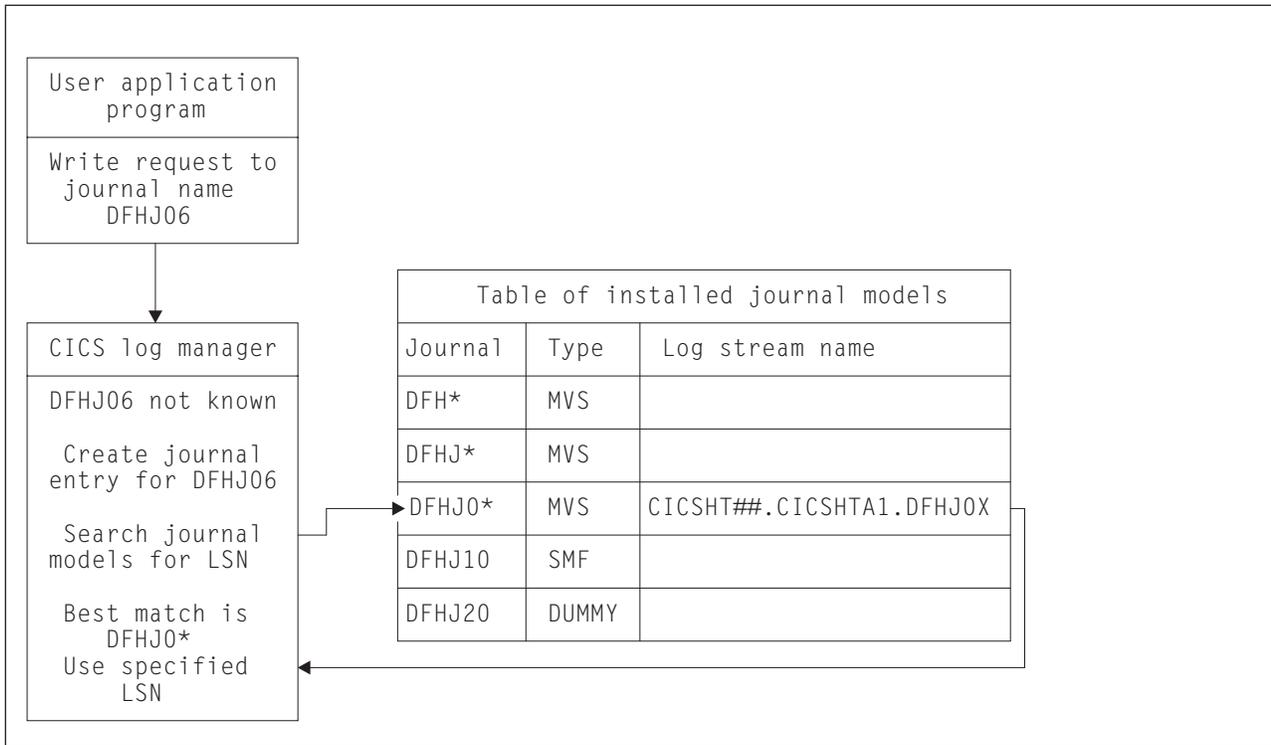


Figure 8. Looking for a journal model that matches a journal name. CICS searches the Table of installed journal models to find the log stream name that corresponds to the journal name, using a “best-match” algorithm.

## Mapping log streams

Except for VSAM RLS forward recovery log stream names, which are obtained directly from the VSAM catalog, CICS maps the system log name or journal name to a corresponding log stream name. CICS does this either by using a user-defined JOURNALMODEL resource definition (if one exists), or by using default names created by resolving symbolic names.

## Mapping of the system log stream

On a cold start, or warm or emergency restart, CICS retrieves the system log stream name from the CICS global catalog.

CICS uses the default log stream names unless you provide a JOURNALMODEL resource definition for the system log.

If there are JOURNALMODEL definitions for your system logs (CICS finds JOURNALMODEL definitions with JOURNALNAME(DFHLOG) and JOURNALNAME(DFHSHUNT)), it attempts to connect to the system log streams named in these definitions. System log stream names must be unique to the CICS region.

If you define JOURNALMODEL resource definitions for your system logs, ensure that:

- The log streams named in the definition are defined to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If there are no suitable JOURNALMODEL definitions, CICS attempts to connect to system log streams, using the following default names:

- userid.applid.DFHLOG
- userid.applid.DFHSHTUNT

where 'userid' is the RACF userid under which the CICS address space is running, and 'applid' is the region's VTAM APPL name. The CSD group DFHLGMOD holds the CICS-supplied JOURNALMODEL definitions for the default DFHLOG and DFHSHTUNT log streams.

Before you try to use these default log stream names, ensure that

- The default log streams are defined explicitly to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If these log streams are not available (perhaps they have not been defined to MVS) or the definitions are not found (perhaps they have not been installed), CICS attempts to create system log streams using a model log stream named *&sysname.LSN\_last\_qualifier.MODEL*, where:

- *&sysname* is the MVS symbol that resolves to the system name of the MVS image
- *LSN\_last\_qualifier* is the last qualifier of the log stream name as specified on the JOURNALMODEL resource definition.

If you do not provide a JOURNALMODEL resource definition for DFHLOG and DFHSHTUNT, or if you use the CICS definitions supplied in group DFHLGMOD, the model names default to *&sysname.DFHLOG.MODEL* and *&sysname.DFHSHTUNT.MODEL*. Once these log streams have been created, CICS then connects to them.

Figure 9 on page 58 shows a graphical representation of the system log mapping process during an INITIAL start.

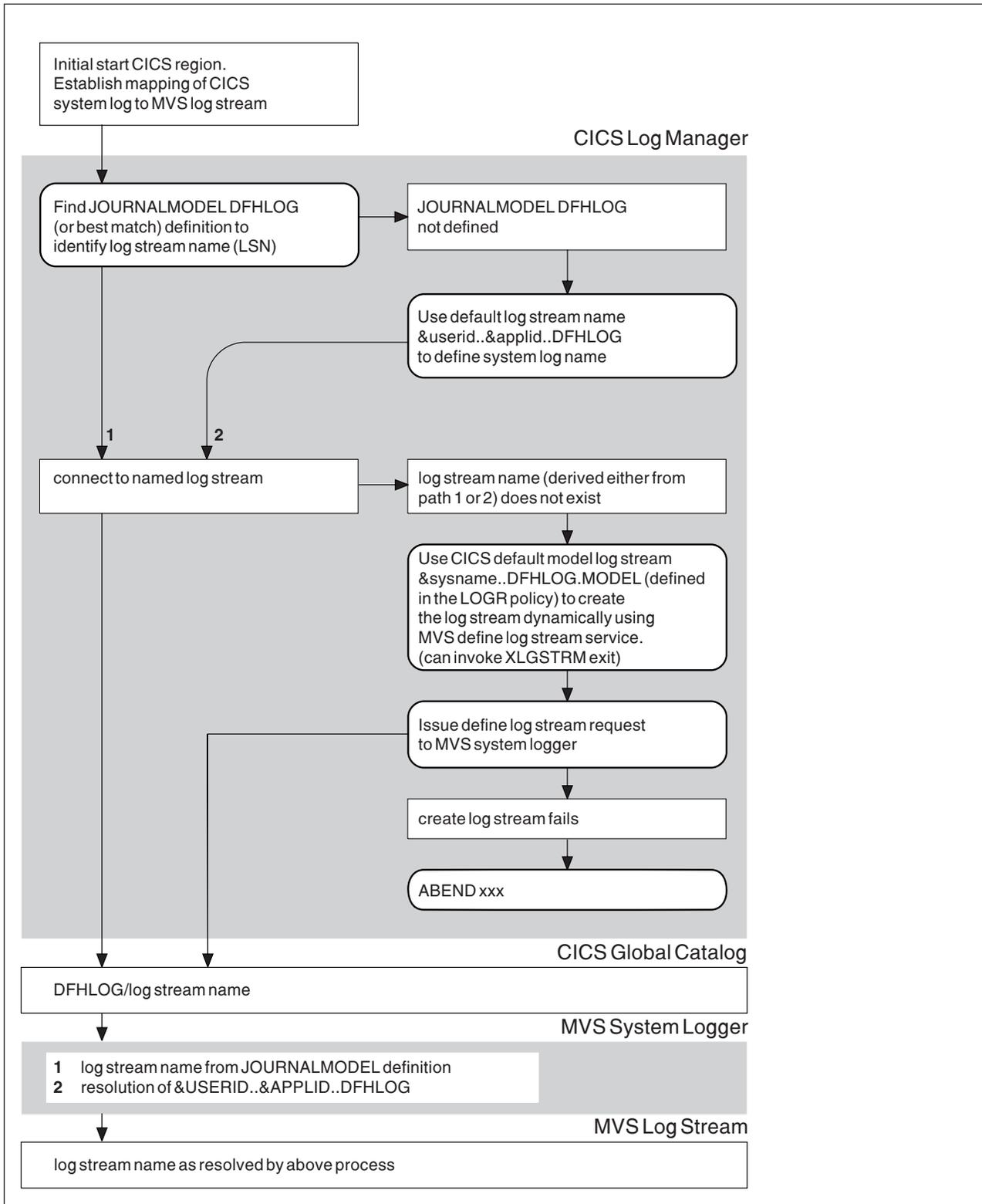


Figure 9. How CICS maps the system log (DFHLOG) to a log stream name (LSN) during an INITIAL start. CICS uses the same process for the secondary system log, DFHSHUNT.

---

## Mapping of general log streams

CICS uses the default log stream names unless you provide a JOURNALMODEL resource definition for the journal or log.

If there is a JOURNALMODEL definition for the log, CICS attempts to connect to the log stream named in the definition.

If you define JOURNALMODEL resource definitions for your system logs, ensure that:

- The log streams named in the definition are defined to the MVS system logger, or
- Suitable model log streams are defined so that they can be created dynamically.

If CICS cannot connect to the log stream named in the JOURNALMODEL definition, it attempts to connect to a log stream, using the default name:

```
userid.applid.journalname
```

Before you try to use this default log stream name, ensure that

- The default log stream is defined explicitly to the MVS system logger, or
- A suitable model log stream is defined so that it can be created dynamically.

If the log stream is not available (perhaps it has not been defined to MVS) or the definition is not found (perhaps it has not been installed), CICS attempts to create a log stream using the default name:

```
LSN_QUALIFIER1.LSN_QUALIFIER2.MODEL
```

where the qualifier fields are based on the JOURNALMODEL definition streamname attribute, as follows:

- If the log stream being created has a qualified name consisting of only two names (*qualifier1.qualifier2*) or has an unqualified name, CICS constructs the model name as *qualifier1.MODEL* or *name.MODEL*.
- If the log stream being created has a qualified name consisting of 3 or more names (*qualifier1.qualifier2....qualifier\_n*), CICS constructs the model name as *qualifier1.qualifier2.MODEL*.

Once the log stream has been created, CICS connects to it.

Figure 10 on page 60 shows a graphical representation of the mapping process for general logs.

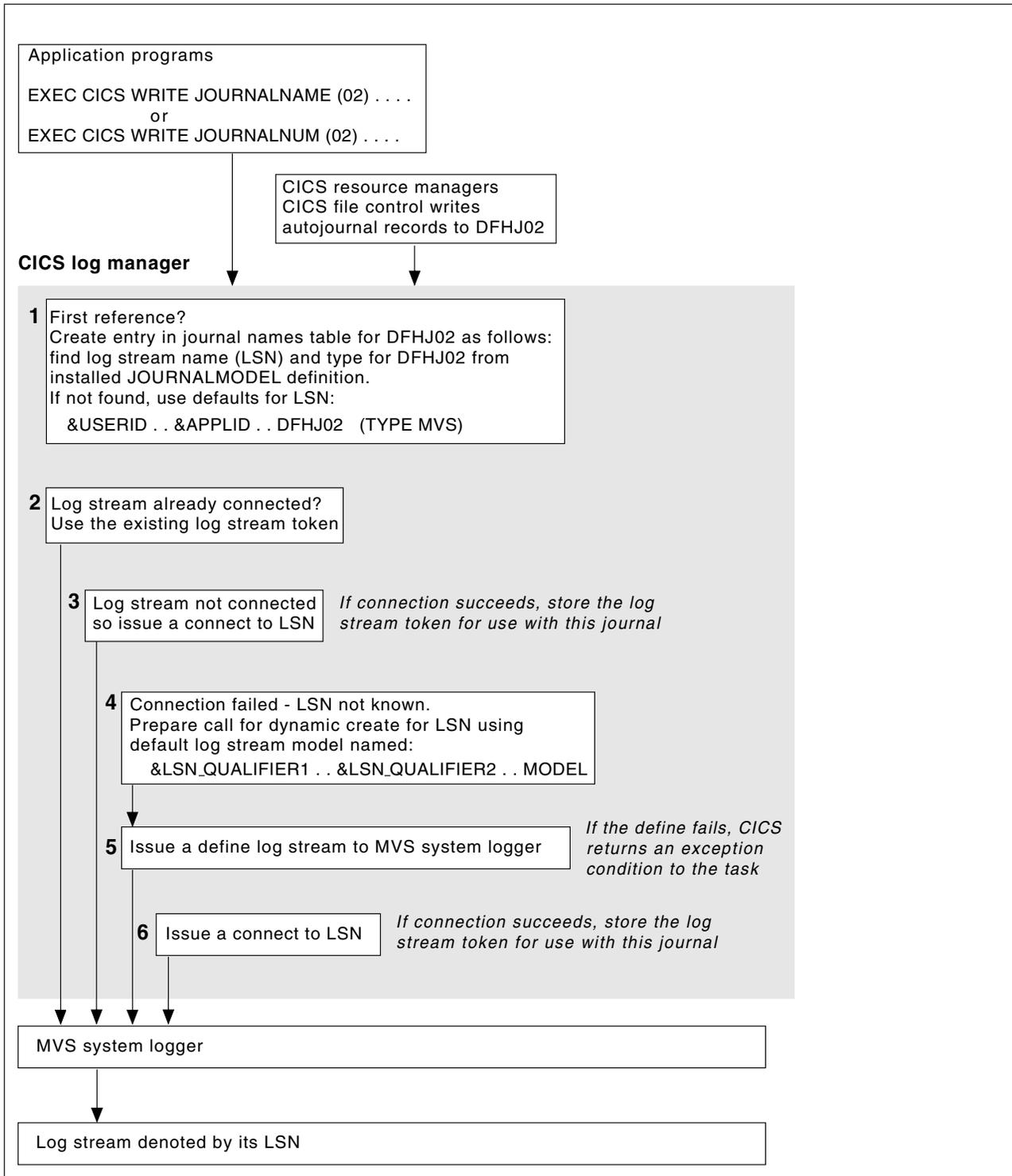


Figure 10. How a CICS journal is mapped to its log stream name (LSN). The name is DFHJ02, used here for user journaling and file control autojournaling.

---

## Using the Journal utility program, DFHJUP

CICS provides a journal utility program, DFHJUP.

You can use the DFHJUP utility program which uses the SUBSYS=(LOGR... facility to select, print, or copy data held on MVS system logger log streams. Alternatively, you can use your own utility to use the SUBSYS=(LOGR... facility.

For information about running DFHJUP, and the SUBSYS=(LOGR.. , facility, see the *CICS Operations and Utilities Guide*.



---

## Chapter 7. Setting up the CICS system definition data set

This chapter describes how to define and initialize a system definition data set (CSD) that CICS needs to store definitions of the resources that it uses.

This chapter also discusses some considerations regarding the use of the CEDA transaction, particularly when a CSD is being shared by more than one CICS region.

You may have used the CEDA transaction already, when running the interactive installation verification procedures (IVPs) after installing CICS. If you ran any of the IVPs (for example, the jobs called DFHIVPBT or DFHIVPOL), you also used a CSD. For information about DFHIVPBT and DFHIVPOL, see the *CICS Transaction Server for z/OS Installation Guide*. Note that the CSD created by the IVPs is limited in size, and initialized with the CICS-supplied resource definitions only.

A CSD is mandatory for some resource definitions. If you are creating a CSD for the first time, go through the steps listed below under “Creating a CSD”: The remainder of this chapter describes these steps in more detail.

If you are already using a CSD with a previous release of CICS, upgrade your CSD to include CICS resource definitions new in CICS Transaction Server for z/OS, Version 3 Release 1. For information about upgrading your CSD, see the *CICS Operations and Utilities Guide*.

You can run the DFHCSDUP offline utility as a batch job to read from and write to the CSD. You should give UPDATE access to the CSD to **only** those users who are permitted to use the DFHCSDUP utility.

---

### Creating a CSD

If you do not already have a CSD in use at your installation:

1. Decide how much disk space you require.
2. Decide whether you want to use the CSD in RLS or non-RLS mode. Having the CSD open in RLS mode allows more than one CICS region to update the CSD concurrently. However, if your CSD is defined as a recoverable data set, and you want update it using the batch utility, DFHCSDUP, you must quiesce the CSD in the CICS regions before running DFHCSDUP.

If you decide to use RLS for the CSD, specify CSDRLS=YES as a system initialization parameter. See “VSAM record-level sharing (RLS)” on page 114.

3. Decide whether the CSD is to be eligible for backup-while-open (BWO<sup>6</sup>). If so, you require the following components of DFSMS 1.2 or later:
  - DFSMShsm
  - DFSMSdss

If the CSD data set is to be eligible for BWO, it must have an ICF catalog entry and be defined in SMS-managed storage. You must also specify:

- CSDBKUP=DYNAMIC as a system initialization parameter for a CSD accessed in non-RLS mode, and for which you have not specified recovery attributes in the ICF catalog.

---

6. Eligibility for BWO means that DFSMS components can back up the CSD while the data set is open for update.

- BWO(TYPECICS) in the ICF catalog for a CSD accessed in RLS mode. You can also specify BWO(TYPECICS) for a CSD accessed in non-RLS mode if you have specified recovery attributes for the data set in the ICF catalog.
4. Define and initialize the CSD.
  5. Decide what CICS file processing attributes you want for your CSD. Although the CSD is a CICS file-control-managed data set, you define file control resource definitions for the CSD by specifying CSDxxxx system initialization parameters (see “Defining CSD attributes” on page 67).
  6. Decide what backup and recovery procedures you require for your CSD.
    - If your CSD is accessed in RLS mode and you decide to make it a recoverable data set, specify the appropriate LOG parameter in the ICF catalog.
    - If your CSD is accessed in non-RLS mode and you decide to make it a recoverable data set, specify the CSDRECOV attribute in the file resource definition, or the appropriate LOG parameter in the ICF catalog entry. If you specify the recovery attribute on the LOG parameter (using AMS DEFINE or ALTER), this overrides the recovery value specified in the file control resource definition.
  7. Decide if you want to use command logs for RDO; see “Logging RDO commands” on page 80 for details of the CADL, CAIL, CRDI, CSDL, CSFL, CSKL, CSPL, and CSRL destinations that CICS uses for RDO command logs.
  8. Make the CSD available to CICS, either by using dynamic allocation or by including the necessary DD statement in the CICS startup job stream. For dynamic allocation of the CSD, you name the fully qualified data set name, and the disposition, on the CSDDSN and the CSDDISP system initialization parameters respectively.
 

When you have started CICS, test the RDO transactions CEDA, CEDB, and CEDC. For information about these transactions, see the *CICS Resource Definition Guide*.
  9. Finally, if you want to restrict access to particular CICS-supplied transactions by applying security, define the necessary transaction profiles to RACF or other external security manager (ESM) and authorize userids as appropriate. For information about how to do this, see the *CICS RACF Security Guide*.

For information about migrating CICS control tables to RDO using the MIGRATE command, see the *CICS Operations and Utilities Guide*.

---

## Calculating CSD disk space

Before you can create the CSD, you must calculate the amount of space you need in your CSD for definition records. Use the following information:

- Each resource definition (for example each program, transaction and terminal) needs one record; the sizes of these definition records are:

Resource	Definition record size (maximum)
Connection	192 bytes
Corbaserver	1187 bytes
DB2CONN	213 bytes
DB2ENTRY	136 bytes
DB2TRAN	90 bytes
DJAR	337 bytes
Doctemplate	195 bytes
Enqmodel	345 bytes

<b>Resource</b>	<b>Definition record size (maximum)</b>
File	322 bytes
Journalmodel	115 bytes
LSR pool	326 bytes
Map set	101 bytes
Partition set	101 bytes
Partner	294 bytes
Processtype	110 bytes
Profile	146 bytes
Program	432 bytes
Requestmodel	1104 bytes
Session	184 bytes
TCPIPSERVICE	242 bytes
Terminal	211 bytes
Tranclass	88 bytes
Transaction	453 bytes
Transient data queue	262 bytes
Tsmodel	201 bytes
Typeterm	368 bytes

- Each group requires two 122-byte records
- Each group list requires two 122-byte records
- Each group name within a list requires one 68-byte record

In your calculation, allow for approximately 1200 CICS-supplied resource definitions of various types, which are loaded into the CSD when you initialize the CSD with the utility program, DFHCSDUP. Finally, add a suitable contingency (approximately 25%), and use your calculated figure when you define the VSAM cluster for the CSD. (See the sample job in Figure 11 on page 66.)

---

## Initializing the CSD

Before you can use the CSD, you must define it as a VSAM KSDS data set, and initialize it using the DFHCSDUP utility program. (See Figure 11 on page 66.)

The INITIALIZE command initializes your CSD with definitions of the CICS-supplied resources. After initialization, you can migrate resource definitions from your CICS control tables, and begin defining your resources interactively with CEDA. You use INITIALIZE only once in the lifetime of the CSD.

The command LIST ALL OBJECTS lists the CICS-supplied resources that are now in the CSD.

```

//DEFINIT JOB accounting information
//DEFCSO EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//AMSDUMP DD SYSOUT=A
//SYSIN DD *
  DEFINE CLUSTER -
    (NAME(CICSTS31.CICS.applid.DFHCSO) -
     VOLUMES(vol1id) -
     KEYS(22 0) -
     INDEXED -
     RECORDS(n1 n2) -
     RECORDSIZE(200 2000) -
     FREESPACE(10 10) -
     SHAREOPTIONS(2) -
     LOG(ALL) -
     LOGSTREAMID(CICSTS31.CICS.CSO.FWDRECOV) -
     BWO(NO)

    DATA -
      (NAME(CICSTS31.CICS.applid.DFHCSO.DATA) -
       CONTROLINTERVALSIZE(8192)) -
    INDEX -
      (NAME(CICSTS31.CICS.applid.DFHCSO.INDEX))
/*
//INIT EXEC PGM=DFHCSO,REGION=300K
//STEPLIB DD DSN=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//DFHCSO DD DSN=CICSTS31.CICS.applid.DFHCSO,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *
      INITIALIZE
      LIST ALL OBJECTS
/*
//

```

Figure 11. Sample job to define and initialize the CSO

#### Notes:

- 1** The key length is 22 bytes, and the KEYS parameter must be coded as shown.
- 2** The average record size of 200 bytes is calculated for a CSO that contains only the CICS-supplied resource definitions (generated by the INITIALIZE and UPGRADE commands). If you create a larger proportion of terminal resource definition entries than are defined in the initial CSO, the average record size is higher because of the larger size of the terminal-type entries. The TERMINAL and TYPETERM definition record sizes are listed under “Calculating CSO disk space” on page 64. The maximum record size must be 2000 as shown.
- 3** Code the SHAREOPTIONS parameter as shown.
- 4** You can specify the recovery attributes for the CSO in the ICF catalog instead of using the CSO system initialization parameters. If you decide to use the CSO in RLS mode, you must define the recovery attributes in the ICF catalog.

You specify the recovery attributes as:

- LOG(NONE) (Nonrecoverable data set)
- LOG(UNDO) (For backout only)
- LOG(ALL) (For both backout and forward recovery)

If you specify LOG(ALL), you must also specify LOGSTREAMID to define the 26-character name of the MVS log stream to be used as the forward recovery log. If you specify recovery attributes in the ICF catalog, and also want to use BWO, specify LOG(ALL) and BWO(TYPECICS).

**5** The DDNAME for the CSD must be DFHCSD.

## Creating a larger CSD

To avoid the CSD filling while CICS is running, ensure that you define the data set with primary and secondary space parameters, and that there is sufficient DASD space available for secondary extents. If your CSD fills up while you are running a CEDA transaction (or the offline utility), define a larger data set and use an AMS command, such as REPRO, to recover the contents of the CSD. If your CSD was dynamically allocated, you can close it, delete it, and redefine it as a larger data set. If your CSD was not dynamically allocated, you must shut down CICS to create a larger data set.

For a description of the commands that you can use for copying files, see the *MVS/ESA Integrated Catalog Administration: Access Method Services Reference* manual.

---

## Defining CSD attributes

File processing attributes for the CSD are defined in the following system initialization parameters:

<b>CSDACC</b>	The type of access allowed.
<b>CSDBKUP</b>	Specify whether the CSD is eligible for BWO. This parameter is ignored if CSDRLS=YES—CICS uses the BWO parameter in the ICF catalog instead. CICS also uses the BWO parameter in the ICF catalog for non-RLS mode CSDs if the LOG parameter in the ICF catalog specifies either UNDO or ALL.
<b>CSDBUFND</b>	The number of buffers for CSD data. Ignored if CSDRLS=YES.
<b>CSDBUFNI</b>	The number of buffers for the CSD index. Ignored if CSDRLS=YES.
<b>CSDDISP</b>	The disposition of the CSD data set.
<b>CSDDSN</b>	The JCL data set name (DSNAME) of the CSD.
<b>CSDFRLOG</b>	A forward recovery journal identifier. This parameter is ignored if CSDRLS=YES, or if the recovery attributes are defined in the ICF catalog on the LOG parameter, in which case LOGSTREAMID from the ICF catalog is used instead.
<b>CSDINTEG</b>	The level of read integrity to be used for a CSD accessed in RLS-mode.
<b>CSDJID</b>	An identifier for automatic journaling.
<b>CSDLSRNO</b>	A VSAM local shared resource pool. Ignored if CSDRLS=YES.
<b>CSDRECOV</b>	Whether or not the CSD is recoverable. This parameter is ignored if CSDRLS=YES and CICS uses the LOG parameter from the ICF catalog instead. If LOG is “undefined”, any attempt to open the CSD in RLS mode fails.  If CSDRLS=NO, this parameter is used only if LOG in the ICF catalog is “undefined.” If LOG in the ICF catalog specifies NONE, UNDO, or ALL, the LOG parameter overrides CSDRECOV.
<b>CSDRLS</b>	Whether the CSD is accessed in RLS or non-RLS mode.
<b>CSDSTRNO</b>	The number of strings for concurrent requests. CSDSTRNO is ignored if CSDRLS=YES and 1024 is assumed.

These parameters are described in greater detail in Chapter 18, “Specifying CICS system initialization parameters,” on page 151.

---

## Sharing the CSD in non-RLS mode

This section describes considerations that affect how you can implement the sharing of a CSD that is accessed in a non-RLS mode, such as LSR or NSR. Sharing the CSD by several CICS regions enables those regions to use the same definitions, and means there is no need for duplicate data sets. This is particularly important in a parallel sysplex environment where a CICSplex may comprise a number of cloned regions, in which case it is essential that they use the same CSD.

To optimize the sharing of a CSD, you should observe the following considerations:

### Shared user access from the same CICS region

- Several users in a CICS region can access the CSD at the same time.
- If you have specified read/write access for the CSD, all the CEDA users in a CICS region can perform read and write functions. CICS file control manages concurrent access for multiple users within a region, using the attributes specified in the CSDACC system initialization parameter.

For more information, see “Multiple users of the CSD within a CICS region (non-RLS)” on page 69.

### Shared user access from several CICS regions

- Several users in different CICS regions can access the CSD at the same time.
- Only one CICS region should be given read/write access to the CSD (CSDACC=READWRITE system initialization parameter). That CICS region should be at the latest level, to ensure that obsolete resource attributes from earlier release can still be updated safely. Other CICS regions should be given only read access to the CSD (CSDACC=READONLY system initialization parameter). This ensures that the CSD integrity is preserved for CICS regions in the same MVS image or different MVS images.
- If you update your shared CSD from one region only, and use CEDA in all the other regions just to install into the required region, specify read/write access for the updating region and read-only for all other regions.
- If you want to update the CSD from several CICS regions, you can use the CICS transaction routing facility, and MRO or ISC, to enable read-only CICS regions to update the CSD. The procedure to follow is:
  1. Select one region that is to own the CSD (the CSD-owning region), and only in this region specify read/write access for the CSD.
  2. Define the CSD as read-only to other CICS regions.
  3. For all regions other than the CSD-owning region:
    - a. Redefine the CEDB transaction as a remote transaction (to be run in the CSD-owning region).
    - b. Install the definition and add the group to your group list for these regions.

You may then use the CEDB transaction from any region to change the contents of the CSD, and use CEDA to INSTALL into the invoking region. You cannot use CEDA to change the CSD in region(s) that do not own the CSD.

If the CSD-owning region fails, the CSD is not available through the CEDB transaction until emergency restart of the CSD-owning region has completed (when any backout processing on the CSD is done). If you try to install a

CSD GROUP or LIST that is the target of backout processing, before emergency restart, you are warned that the GROUP or LIST is internally locked to another user. Do not run an offline VERIFY in this situation, because backout processing removes the internal lock when emergency restart is invoked in the CSD-owning region.

If you do not want to use the above method, but still want the CSD to be defined as a recoverable resource, then integrity of the CSD cannot be guaranteed. In this case, you must not specify CSDBKUP=DYNAMIC, because the CSD would not be suitable for BWO.

- You can define several CICS regions with read/write access to the CSD, but this should only be considered if the CICS regions run in the same MVS image, and all are at the latest CICS level.
- If you give several CICS regions read/write access to the same CSD, and those regions are in the same MVS image, integrity of the CSD is maintained by the SHAREOPTIONS(2) operand of the VSAM definition, as shown in the Figure 11 on page 66.
- If you give several CICS regions read/write access to the same CSD, and those regions are in different MVS images, the VSAM SHAREOPTIONS(2) operand does not provide CSD integrity, because the VSAMs for those MVS images do not know about each other.

For more information about shared CSD access within one MVS image, see “Sharing a CSD by CICS regions within a single MVS image (non-RLS)” on page 70. For more information about shared CSD access across several MVS images, see “Sharing a CSD in a multi-MVS environment (non-RLS)” on page 71. For information about sharing the CSD between different releases of CICS, see “Sharing the CSD between different releases of CICS” on page 72.

**Shared access from CICS regions and DFHCSDUP:** If you want to use the DFHCSDUP utility program in read/write mode to update the CSD, you must ensure that no CICS users are using any of the CEDA, CEDB, or CEDC transactions.

For information about other factors that can restrict access to a CSD, see “Other factors restricting CSD access” on page 73.

For information about the system initialization parameters for controlling access to the CSD, see “Defining CSD attributes” on page 67.

## Multiple users of the CSD within a CICS region (non-RLS)

If you have specified read/write access for the CSD, all the CEDA users in a CICS region can perform read and write functions. CICS file control manages concurrent access for multiple users within a region, using the attributes specified in the CSDACC system initialization parameter.

CICS protects individual resource definitions against concurrent updates by a series of internal locks on the CSD. CICS applies these locks at the group level. While CICS is executing a command that updates any element in a group, it uses the internal lock to prevent other RDO transactions within the region from updating the same group. CICS removes the lock record when the updating command completes execution. Operations on lists are also protected in this way.

The number of concurrent requests that may be processed against the CSD is defined by the CSDSTRNO system initialization parameter. Each user of CEDA (or

CEDB or CEDC) requires two strings, so calculate the CSDSTRNO value by first estimating the number of users that may require concurrent access to the CSD, and then multiply the number by two.

CEDA issues a diagnostic message if the CSDSTRNO value is too small to satisfy the instantaneous demand on the CSD for concurrent requests. A subsequent attempt to reissue the command succeeds if the conflict has disappeared. If conflicts continue to occur, increase the CSDSTRNO value.

## Sharing a CSD by CICS regions within a single MVS image (non-RLS)

The CSD may be shared by a number of CICS regions within the same MVS image. You can maintain the integrity of the CSD in this situation by coding SHAREOPTIONS(2) on the VSAM definition, as shown in the Figure 11 on page 66. The CICS attributes of the CSD, as viewed by a given region, are defined in the system initialization parameters for that region.

You should consider defining:

- One CICS region with read/write access (CSDACC=READWRITE) to the CSD. That region can use all the functions of CEDA, CEDB, and CEDC.
- Other CICS regions with only read access (CSDACC=READONLY) to the CSD. Such CICS regions can use the CEDC transaction, and those functions of CEDA and CEDB that do not require write access to the CSD (for example, they can use INSTALL, EXPAND, and VIEW, but not DEFINE). You can enable such CICS regions to update the CSD, by using the procedure described on page 68.

**Note:** Read integrity is not guaranteed in a CICS region that has read-only access to a shared CSD. For example, if one CICS region that has full read/write access updates a shared CSD with new or changed definitions, another CICS region with read-only access might not obtain the updated information. This could happen if a control interval (CI) already held by a read-only region (before an update by a read/write region) is the same CI needed by the read-only region to obtain the updated definitions. In this situation, VSAM does not reread the data set, because it already holds the CI. However, you can minimize this VSAM restriction by specifying CSDLSRNO=NONE, and the minimum values for CSDBUFNI and CSDBUFND, but at the expense of degraded performance. See “Specifying read integrity for the CSD” on page 75 for information about read integrity in a data set accessed in RLS mode.

If you define several CICS regions with read/write access to the CSD, those regions should all be at the latest level. Only one CICS region with read/write access can use a CEDA, CEDB, or CEDC transaction to access the CSD, because the VSAM SHAREOPTIONS(2) definition prevents other regions from opening the CSD.

If you are running CICS with the CSD defined as a recoverable resource (CSDRECOV=ALL), see “Planning for backup and recovery” on page 76 for some special considerations.

You can use CEMT to change the file access attributes of the CSD, or you can use the EXEC CICS SET FILE command in an application program. However, ensure that the resulting attributes are at least equivalent to those defined either by CSDACC=READWRITE or CSDACC=READONLY. These parameters allow the following operations on the CSD:

**READONLY** Read and browse.  
**READWRITE** Add, delete, update, read and browse.

## Sharing a CSD in a multi-MVS environment (non-RLS)

If you need to share a CSD between CICS regions that are running in different MVS images, you should ensure that only one region has read/write access.

The VSAM SHAREOPTIONS(2) offers no integrity, because the VSAMs running in a multi-MVS environment do not know of each other.

Because of these limitations, active and alternate CICS regions running in different MVS images must not share the CSD with other CICS regions, unless you are using some form of global enqueueing (for example, with global resource serialization (GRS)).

These multi-MVS restrictions also apply to running the offline utility, DFHCSDUP.

## Multiple users of one CSD across CICS or batch regions (non-RLS)

The types of access needed in the four situations where the CSD are used are shown in Table 11.

Table 11. CSD access

	Type of activity	Access
1	CICS region performing initialization (cold or initial start)	Read-only
2	CICS region running one or more CEDA, CEDB, or CEDC transactions	Read/write or read-only (as specified in the CSDACC parameter)
3	Batch region running utility program DFHCSDUP	Read/write or read only, depending on PARM parameter
4	CICS regions performing emergency restart, and CSD file backout is required	Read/write

Note the following limitations when the activities listed in Table 11 are attempted concurrently:

1. You cannot run DFHCSDUP in read/write mode in a batch region if any CICS region using the same CSD is running one of the CEDA, CEDB, or CEDC transactions. (The exception is when the CEDx transactions accessing the CSD are in a region (or regions) for which the CSD is defined as read-only.)
2. None of the CEDx transactions runs if the CSD to be used is being accessed by the DFHCSDUP utility program in read/write mode. (This restriction does not apply if the transaction is run in a region for which the CSD is defined as read-only.)
3. None of the CEDx transactions runs in a CICS region whose CSD is defined for read-write access if any of the RDO transactions are running in another CICS region that has the CSD defined for read-write access.

A CICS region starting with an initial or cold start opens the CSD for read access only during initialization, regardless of the CSDACC operand. This enables a CICS region to be initialized even if a user on another region or the DFHCSDUP utility program is updating the CSD at the same time. After the group lists are installed, CICS leaves the CSD in a closed state.

On a warm or emergency start, the CSD is not opened at all during CICS initialization if CSDRECOV=NONE is coded as a system initialization parameter.

However, if CSDRECOV=ALL is coded, and backout processing is pending on the CSD, the CSD is opened during CICS initialization on an emergency start.

## Sharing the CSD between different releases of CICS

Resource attributes become obsolete when they have no relevance for a new release of CICS. CICS continues to display these on CEDx panels, but they are displayed as protected fields, indicating that they are not supported by this release. Using the ALTER command on definitions that specify obsolete attributes does not cause the loss of these attributes, so you can safely update resource definitions using this release. If you are sharing the CSD with CICS regions at an earlier release, you can update the unsupported fields by using the PF2 function key to remove the protection when in ALTER mode. (PF2 is designated as the “compatibility” key (COM) on the CEDA or CEDB display panels.) Pressing PF2 converts protected fields to unprotected fields that you can modify. If you want to use this facility to enable you to share common resource definitions, the rule for sharing between different release levels of CICS is that you must update the CSD from the highest release level.

For information about using the CEDA and CEDB ALTER commands to update resource definitions in compatibility mode, see the *CICS Resource Definition Guide*.

You can also use the CSD utility program, DFHCSDUP, to update resources that specify obsolete attributes. A compatibility option is added for this purpose, which you must specify on the PARM parameter on the EXEC PGM=DFHCSDUP statement. You indicate the compatibility option by specifying COMPAT or NOCOMPAT. The default is NOCOMPAT, which means that you cannot update obsolete attributes.

### CICS regions that use DB2

If you share your CSD between different releases of CICS that use DB2, you must use the DB2 resource definitions appropriate for each release of CICS. With those releases of CICS that ship the CICS DB2 attachment facility, you must use the CICS-supplied group called DFHDB2. This group is included in the CICS-supplied startup list, DFHLIST, and specifies different program names from the attachment facility provided by DB2.

For earlier releases of CICS that do not provide the DFHDB2 group, you must use your own resource definitions that specify the resource names appropriate for the release of CICS and DB2.

### CICS-supplied compatibility groups

If you are sharing the CSD between CICS Transaction Server for z/OS, Version 3 Release 1 and an earlier release of CICS, you must ensure that the group list you specify (on the GRPLIST system initialization parameter) contains all the CICS-required standard definitions. When you upgrade the CSD to the CICS Transaction Server for z/OS, Version 3 Release 1 level, some of the IBM groups referenced by your group list are deleted, and the contents transferred to one of the compatibility groups, DFHCOMPx. To ensure that these continue to be available to your CICS regions of earlier releases, add the compatibility groups *after* all the other CICS-supplied definitions.

For information about upgrading your CSD, and about the compatibility groups in CICS Transaction Server for z/OS, Version 3 Release 1, see the *CICS Resource Definition Guide*.

## Other factors restricting CSD access

Access to the CSD may also be restricted if it is left open after abnormal termination of a CEDA, CEDB, or CEDC transaction. If the CSD is left open with write access, this prevents other address spaces from subsequently opening it for write access. This situation can be remedied by using CEMT to correct the status of the CSD.

Access to the CSD is not released until the RDO transaction using it is ended, so users of CEDA, CEDB, and CEDC should ensure that a terminal running any of these transactions is not left unattended. Always end the transaction with PF3 as soon as possible. Otherwise, users in other regions are unable to open the CSD.

There may be times when you cannot create definitions in a group or list. This situation arises if an internal lock record exists for the group or list you are trying to update. If you are running the DFHCSDUP utility program (or a CEDA transaction) when this occurs, CICS issues a message indicating that the group or list is locked. As described under “Multiple users of the CSD within a CICS region (non-RLS)” on page 69, this is normally a transient situation while another user within the same region is updating the same group or list. However, if a failure occurs, preventing a CEDA transaction from completing successfully, and CSDRECOV=NONE is coded, the internal lock is not removed and is left in force. (If CSDRECOV=ALL is coded, the CSD is recoverable and file backout occurs and frees the lock.) This could happen, for example, if a system failure occurs while a CEDA transaction is running; it could also happen if the CSD becomes full. You can remedy this situation by running the DFHCSDUP utility program with the VERIFY command.

However, if you have coded CSDRECOV=ALL, make sure no backout processing is pending on the CSD before you run an offline VERIFY. The effect of coding CSDRECOV=ALL is discussed more fully under “Planning for backup and recovery” on page 76.

---

## Sharing the CSD in RLS mode

This section discusses the use of VSAM RLS to enable the CSD to be shared between a number of CICS regions. The reasons for, and benefits of, sharing the CSD are the same regardless of the access mode. However, there are some RLS-related factors that you need to consider if you decide to operate CICS with the CSD in RLS mode.

The following requirements and rules apply to using the CSD in RLS-mode:

- Your CICS regions must run in an RLS-capable environment. That is, all the CICS regions must reside in a parallel sysplex, and an SMSVSAM server must be running in each MVS image that supports one or more CICS regions.
- The CSD must reside in SMS-managed storage.
- You must specify CSDRLS=YES in all CICS regions that are sharing the CSD in RLS-mode, and RLS must be enabled in each region (by the RLS=YES system initialization parameter).
- As soon as the first CICS region opens the CSD in RLS mode, it can only be opened in RLS mode by other CICS regions. If a CICS region attempts to open the CSD in non-RLS mode when it is open in RLS mode by other regions, the non-RLS open request fails.

**Note:** This rule means that you cannot use a CSD in RLS mode on a CICS release that supports RLS and share it with CICS regions that do not support RLS. The sharing by non-RLS capable regions means that a CSD can only be use in non-RLS mode.

- All the rules governing the use of a data set in RLS mode apply also to the CSD—there are no special rules for the CSD because it is a CICS system data set.
- Any number of CICS regions can open the CSD in RLS mode and all can use CEDA to update the data set with full integrity. The CICS regions can reside in different MVS images, but the MVS images must be in the same sysplex. There is no need to restrict updating to only one CICS region as in the case of non-RLS sharing, and you can specify the CSDACC=READWRITE system initialization parameter for all CICS regions that specify CSDRLS=YES.

## Differences in CSD management between RLS and non-RLS access

Although a CSD accessed in RLS mode is protected by VSAM RLS locking, this operates at the CICS file control level. It does not change the way the CEDA and CEDB transactions manage the integrity of CSD groups.

The CEDx transactions protect resource definitions in the same way for RLS mode and non-RLS mode CSDs. They protect individual resource definitions against concurrent updates by a series of internal locks on the CSD. The RDO transactions apply these locks at the group level. While RDO transactions are executing a command that updates any element in a group, they use the internal lock to prevent other RDO transactions within a CICS region from updating the same group. The locks are freed only when the updating command completes execution. Operations on lists are protected in the same way. However, in an RLS environment, these internal locks affect all CICS regions that open the CSD in RLS mode. In the non-RLS case they apply only to the CICS region that has the data set open for update (which can only be a single region).

The use of a single buffer pool by the SMSVSAM server removes some of the problems of sharing data that you get with a non-RLS CSD.

Some other points to note are:

- If a CSD is defined with CSDACC=READWRITE and CSDRLS=YES, more than one CICS region can open the CSD concurrently. However, file control closes the CSD opened in RLS mode at termination of each CEDx transaction, in the same way as for a non-RLS CSD. CSDACC=READONLY is not necessary for a CSD accessed in RLS mode.
- The number of concurrent requests that can be processed against the CSD, is always 1024 for RLS. Diagnostic messages about CSDSTRNO value do not occur for RLS-mode CSDs.
- The VSAM cluster definition SHAREOPTIONS parameter is ignored by SMSVSAM when an application, such as CICS, opens a data set in RLS mode.
- A CSD accessed in RLS mode could give rise to RDO transaction failures that do not occur for a non-RLS mode CSD: For example:
  - An RDO transaction could be holding an RLS exclusive lock while it updates a record, which causes another RDO transaction to time out.
  - If the CSD is recoverable and CICS or MVS fails, update locks of failed-inflight RDO transactions are converted into retained locks. This could result an RDO transaction receiving a LOCKED response from VSAM which, in turn, would cause the RDO transaction to fail.

## Specifying read integrity for the CSD

You can specify that you want read integrity for a CSD opened in RLS mode. This ensures that CEDx INSTALL command always installs the latest version of a resource definition. The CEDA INSTALL command has to wait for a lock on any CSD record that it is trying to install if another CEDx transaction is updating the record. The install completes only when the updating task has finished updating the record and released its exclusive lock.

Although the CSDINTEG system initialization parameter supports both consistent and repeatable read integrity, consistent read integrity should provide all the benefit you need for your RDO operations.

## Specifying file control attributes for the CSD

You specify file control attributes for the CSD using the CSDxxxxx system initialization parameters (see “Defining CSD attributes” on page 67) except for the following:

<b>CSDBKUP</b>	You specify backup-while-open support for the CSD using the VSAM BWO parameter in the ICF catalog.
<b>CSDBUFND</b>	Ignored.
<b>CSDBUFNI</b>	Ignored.
<b>CSDFRLOG</b>	You specify the forward recovery log stream for the CSD using the VSAM LOGSTREAMID parameter in the ICF catalog.
<b>CSDINTEG</b>	You specify read integrity for RDO transactions (CEDx) using this system initialization parameter.
<b>CSDLSRNO</b>	Ignored.
<b>CSDRECOV</b>	You specify the recovery attributes for the CSD using the VSAM LOG parameter in the ICF catalog. If LOG is “undefined”, any attempt to open the CSD in RLS mode will fail.
<b>CSDSTRNO</b>	For RLS the number of strings defaults to 1024.

## Effect of RLS on the CSD batch utility DFHCSDUP

You can use DFHCSDUP to update a **non-recoverable** CSD in RLS mode while CICS also has the CSD open for update in RLS mode. To enable DFHCSDUP to update the CSD in RLS mode, specify RLS=NRI or RLS=CR in the DD statement for the CSD in the DFHCSDUP JCL. Generally, DFHCSDUP does not perform as well in RLS mode as in non-RLS mode.

You cannot run DFHCSDUP while CICS regions have the CSD open in RLS mode if the CSD is defined as recoverable. This is because a non-CICS job, such as DFHCSDUP, is not allowed to open a recoverable data set for output in non-RLS mode while it is already open in RLS mode. Therefore, before you can run DFHCSDUP, you must quiesce the CSD by issuing a CEMT, or an EXEC CICS, SET DATASET(...) QUIESCED command.

A recoverable CSD is unavailable to all CICS regions while DFHCSDUP is running until it is unquiesced, which makes it available again in RLS mode. To unquiesce the CSD at the end of the DFHCSDUP run, issue a CEMT, or an EXEC CICS, DATASET(...) UNQUIESCED command.

For a recoverable CSD, the main factor to consider when planning whether to use RLS is how much you use DFHCSDUP compared with the CEDx transactions. If you use DFHCSDUP frequently to update your production CSD, you may decide that it is better to use the CSD in non-RLS mode. On the other hand, if you use DFHCSDUP only occasionally, and you want the ability to update the CSD online from any CICS region, use RLS.

---

## Planning for backup and recovery

To guard against system failures that affect your CSD, take a backup of the CSD at regular intervals. Then, if the CSD is corrupted for any reason, you can restore it to its state at the last backup. To keep the backup of the CSD as up-to-date as possible, make an image copy of your CSD before each period of update activity, either by an RDO transaction or DFHCSDUP.

Alternatively, because the CSD is open for update whenever RDO work is taking place, it is a good candidate for eligibility for BWO. If the CSD is specified as eligible for BWO, and the data set is corrupted, you can restore a BWO image of the CSD using DFSMSdss, then run forward recovery to the point of corruption using a forward recovery utility.

For a CSD opened in RLS mode, the recovery attributes must be defined in the ICF catalog entry for the CSD, and CICS uses the forward recovery log's log stream name (LSN) from the ICF catalog.

For a CSD opened in non-RLS mode, the recovery attributes can be defined in the ICF catalog entry for the CSD, or on the CSD system initialization parameters. The forward recovery log's log stream name (LSN) is retrieved from either CSDFRLOG or the ICF catalog. If LOG is defined in the catalog, the forward recovery log stream specified in the catalog is used. If LOG is not defined, the CSDFRLOG journal id is used to determine the log stream name.

For a CSD opened in non-RLS mode, you can use the system initialization parameter CSDBKUP=DYNAMIC|STATIC to indicate whether the CSD is eligible for BWO. Specify CSDBKUP=DYNAMIC for BWO support, or STATIC (the default) for a "normal" quiesced backup. If you specify BWO support for the CSD you must also define it as forward recoverable. For more information about BWO, see "Backup while open (BWO) of VSAM files" on page 31.

For a CSD opened in RLS mode, you must specify all recovery attributes, which includes backup, in the ICF catalog. BWO backup eligibility is specified using BWO(TYPECICS).

If you specify forward recovery for the CSD, changes (after images) made by CICS to the CSD are logged in the forward recovery log stream. Using the latest backup, and the after images from forward recovery log stream, you can recover all the changes made by running a recovery program, such as the CICS VSAM forward recovery utility. After performing forward recovery, you must reenter any CEDA transactions that were running at the time of failure, as these are effectively backed out by the forward recovery process. You can find details of these in the CSDL transient data destination, which is the log for copies of all CEDA commands. See "Logging RDO commands" on page 80 for more information.

Recoverability, forward recovery log stream names, and BWO eligibility can be defined optionally in the ICF catalog for a non-RLS accessed CSD, but must be defined the ICF catalog if the CSD is accessed in RLS mode.

The CSDBKUP, CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. Table 12 on page 77 and Table 13 on page 77 summarize their effects when the SIT is assembled and during CICS override processing, respectively.

Table 12. CSDBKUP and related parameters at SIT assembly time for CSDRLS=NO

CSDRECOV	CSDFRLOG	CSDBKUP	Result
ALL	FRLOG from 01 through 99.	Either DYNAMIC or STATIC	OK
ALL	NO	Either DYNAMIC or STATIC	SIT assembly fails with MNOTE stating that CSDRECOV=ALL implies that the CSDFRLOG option must be specified.
BACKOUTONLY or NONE	FRLOG from 01 through 99.	DYNAMIC	SIT assembly fails with assembler MNOTES stating that CSDBKUP=DYNAMIC requires CSDRECOV=ALL and that CSDFRLOG requires CSDRECOV=ALL.
BACKOUTONLY or NONE	NO	DYNAMIC	SIT assembly fails with an assembler MNOTE stating that CSDBKUP=DYNAMIC requires CSDRECOV=ALL.
BACKOUTONLY or NONE	NO	STATIC	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	STATIC	SIT assembly warning MNOTE stating that CSDFRLOG is ignored unless CSDRECOV=ALL.

**Notes:**

1. When CSDBKUP=DYNAMIC, the CSD is eligible for BWO.
2. Backup and recovery attributes must be specified in the ICF catalog for a CSD opened in RLS mode (CSDRLS=YES).
3. Backup and recovery attributes can optionally be specified in the ICF catalog for a CSD opened in non-RLS mode (CSDRLS=NO), but you must still have a consistent set of parameters as defined in the table above.

Table 13. CSDBKUP and related system initialization parameters during CICS override processing (CSDRLS=NO)

CSDRECOV	CSDFRLOG	CSDBKUP (see Notes)	Result
ALL	FRLOG from 01 through 99.	Either DYNAMIC or STATIC.	OK
ALL	NO	Either DYNAMIC or STATIC.	Message DFHPA1944 is issued stating that CSDRECOV=ALL cannot be specified without a CSDFRLOG if CSDRLS=NO. CICS initialization is terminated.
BACKOUTONLY or NONE	FRLOG from 01 through 99.	DYNAMIC	Processing continues and messages DFHPA1929, stating that CSDBKUP has defaulted to STATIC, and DFHPA1930, stating that CSDFRLOG has been ignored, are issued.

Table 13. CSDBKUP and related system initialization parameters during CICS override processing (CSDRLS=NO) (continued)

CSDRECOV	CSDFRLOG	CSDBKUP (see Notes)	Result
BACKOUTONLY or NONE	NO	DYNAMIC	Processing continues and message DFHPA1929 is issued, stating that CSDBKUP has defaulted to STATIC.
BACKOUTONLY or NONE	NO	STATIC	OK
BACKOUTONLY or NONE	FRLOG from 01 through 99.	STATIC	Processing continues and message DFHPA1930 stating that CSDFRLOG has been ignored is issued.

**Notes:**

1. When CSDBKUP=DYNAMIC, the CSD is eligible for BWO.
2. Backup and recovery attributes must be specified in the ICF catalog for a CSD opened in RLS mode (CSDRLS=YES).
3. Backup and recovery attributes can optionally be specified in the ICF catalog for a CSD opened in non-RLS mode (CSDRLS=NO), but you must still have a consistent set of parameters as defined in the table above.

Write and test procedures for backing up and recovering your CSD before beginning to operate a production CICS region.

Forward recovery of the CSD is not possible if CSD updates are made outside CICS. To enable recovery of the updates made outside CICS, you need to use an image copy. If you update the CSD from outside CICS, do not use CEDA to update the CSD until an image copy has been made.

## Transaction backout during emergency restart

If you define the CSD as a recoverable resource, by coding the CSDRECOV system initialization parameter, the same rules apply to the CSD as to any other CICS recoverable resource. If you code CSDRECOV=ALL (or BACKOUTONLY) as a system initialization parameter, and have to perform an emergency restart following a failure, CICS backs out any incomplete RDO transactions that were in-flight at the time of failure.

## Dynamic backout for transactions

CICS performs dynamic transaction backout for any RDO transaction abends. You cannot decide whether you want dynamic transaction backout by coding an attribute on transaction definitions in the CSD; CICS assumes this requirement for all transactions. (Defining the CSD as non-recoverable has the effect of preventing backout, but this is not recommended.)

## Other recovery considerations

When you are deciding what recoverability options to specify, consider the following factors:

- CEDA command syncpoint criteria
- Sharing the CSD with another CICS region
- Accessing the CSD by the offline utility program DFHCSDUP

For information about CEDA command syncpoint criteria, see “CEDA command syncpoint criteria.” For information about sharing the CSD between CICS regions, see “Sharing the CSD in non-RLS mode” on page 68. For information about using the DFHCSDUP utility to access the CSD, see “Accessing the CSD by the offline utility program, DFHCSDUP.”

### **CEDA command syncpoint criteria**

You can issue CEDA in two ways:

1. A single command entered on the command line
2. A series of single commands from within an EXPAND or DISPLAY panel

Commands that change the contents of the CSD commit or back out changes at the single command level. The exception to this rule is a generic ALTER command. A generic ALTER command is committed or backed out at the single resource level.

The replacement of an existing resource definition by an INSTALL command only occurs if the resource is not in use. If any of the resources in the group being installed are in use, the install will fail.

Changes made to the following resource definitions by an INSTALL command are committed at the resource level and are not backed out if the install fails:

AUTOINSTALL MODEL,FILE, LSRPOOL, MAPSET, PARTITIONSET, PARTNER, PROFILE, PROGRAM, TDQUEUE, and TRANSACTION,

Changes made to the following resource definitions by an INSTALL command are committed at the group level and are backed out if the install fails:

CONNECTION, SESSION, TERMINAL, and TYPETERM

### **Accessing the CSD by the offline utility program, DFHCSDUP**

Changes made to the CSD by the offline utility program DFHCSDUP are not recoverable. Also consider the effects of using commands provided by this program, before emergency restart of a failing CICS region, that:

1. Change the contents of lists or groups that are the target of backout processing.
2. Remove internal locks (by using VERIFY, for example).

These situations are analogous to the problems met when using multiple read/write regions, and are discussed above.

---

## Logging RDO commands

If you want to record RDO commands, create definitions for the extrapartition queues CADL, CAIL, CRDI, CSDL, CSFL, CSKL, CSPL, and CSRL. These are used as follows:

<b>CADL</b>	Logs VTAM resources installed in the active CICS region. CICS records in this log all terminal entries installed in the TCT, entries deleted from the TCT, and dynamically installed entries that are discarded. This log includes autoinstalled terminal definitions, terminal definitions installed explicitly by the CEDA INSTALL command, and terminal definitions installed from a group list during system initialization.
<b>CAIL</b>	Logs autoinstall terminal model entries installed in the TCT, and entries deleted from the TCT.
<b>CRDI</b>	Logs installed resource definitions of programs, transactions, mapsets, profiles, partition sets, files, and LSR pools.
<b>CSDL</b>	Logs RDO commands that affect the CSD.
<b>CSFL</b>	Logs file resources installed in the active CICS region. That is, all file entries installed in the FCT, entries deleted from the FCT, dynamically installed entries that are discarded, and messages from dynamic allocation of data sets and from loading CICS data tables.
<b>CSKL</b>	Logs transaction and profile resources installed in the active CICS region. That is, all transaction and profile entries installed in the PCT, entries deleted from the PCT, and dynamically installed entries that are discarded.
<b>CSPL</b>	Logs program resources installed in the active CICS region. That is, all program entries installed in the PPT, entries deleted from the PPT, and dynamically installed entries that are discarded.
<b>CSRL</b>	Logs changes to the set of partner resources installed in the active CICS region. That is, all operations that install or discard partner resources.

If you want these RDO command logs sent to the same destination (CSSL) as the messages, you can use the definitions shown in Figure 12 on page 81. If you like, you can direct these logs to any other transient data queue, or define them as extrapartition data sets.

```

*
DEFINE TDQUEUE (CSSL)          GROUP(DFHDCTG)
    DESCRIPTION(USED FOR MESSAGES)
    TYPE(EXTRA)                 TYPEFILE(OUTPUT)
    RECORDSIZE(132)             BLOCKSIZE(136)
    RECORDFORMAT(VARIABLE)     BLOCKFORMAT(UNBLOCKED)
                                DDNAME(MSGUSR)

*
DEFINE TDQUEUE (CADL)          GROUP(DFHDCTG)
    DESCRIPTION(CEDA VTAM RESOURCE LOGGING)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CAIL)          GROUP(DFHDCTG)
    DESCRIPTION(AITM MESSAGES)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CRDI)          GROUP(DFHDCTG)
    DESCRIPTION(RDO INSTALL LOG)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CSDL)          GROUP(DFHDCTG)
    DESCRIPTION(CEDA COMMAND LOGGING)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CSFL)          GROUP(DFHDCTG)
    DESCRIPTION(FILE ALLOCATION MESSAGES)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CSKL)          GROUP(DFHDCTG)
    DESCRIPTION(TRANSACTION MANAGER MESSAGES)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CSPL)          GROUP(DFHDCTG)
    DESCRIPTION(PROGRAM MANAGER MESSAGES)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

*
DEFINE TDQUEUE (CSRL)          GROUP(DFHDCTG)
    DESCRIPTION(PARTNER RESOURCE MANAGER)
    TYPE(INDIRECT)             INDIRECTNAME(CSSL)

```

Figure 12. Definitions for RDO command logs sent to CSSL

## Making the CSD available to CICS

To make your CSD available to CICS, you can either include a DD statement in the CICS startup job or use dynamic allocation.

- You can include the following DD statement in your CICS startup job stream:

```
//DFHCSD DD DSN=CICSTS31.CICS.applid.DFHCSD,DISP=SHR
```

You usually need the CSD DD statement to include DISP=SHR. (See “Sharing the CSD in non-RLS mode” on page 68.)

If you include a DD statement for the CSD in the CICS startup job, the CSD is allocated at the time of CICS job step initiation, and **remains allocated for the duration of the CICS job step**.

- You may prefer to take advantage of the CICS dynamic allocation of the CSD. If so, do **not** provide a DD statement for the CSD in the startup job stream. If there is a CSD DD statement, it is used instead of dynamic allocation. To dynamically allocate the CSD, specify the data set name (DSNAME) and disposition (DISP) of the CSD, using one of the following methods:

- The CSDDSN and CSDDISP system initialization parameters
- The CEMT SET FILE command
- The EXEC CICS SET FILE command

CICS then uses the full data set name (DSNAME) to allocate the CSD as part of OPEN processing. The CSD is automatically deallocated when the last entry associated with it is closed.

For more information about OPEN processing, see Chapter 12, “Defining user files,” on page 111. For information about the parameters that you can code for the CSD in the SIT, see Chapter 18, “Specifying CICS system initialization parameters,” on page 151.

## Installing the RDO transactions

The RDO transactions, CEDA, CEDB, and CEDC are defined in the CICS-supplied group, DFHSPI. This group is also included in DFHLIST, the CICS group list. Ensure that a copy of DFHSPI is included in the group list that you use for your CICS startup. You specify the group list on the GRPLIST system initialization parameter.

For information about the CEDA, CEDB, and CEDC transactions, see the *CICS Resource Definition Guide*.

## Moving your CICS tables to the CSD

When you have created a CSD, and initialized it with CICS-supplied definitions, you are ready to move the contents of your CICS tables to the CSD data set. You do this by running the offline utility, DFHCSDUP, using the MIGRATE command to specify the table you want to migrate. For information about the DFHCSDUP utility program and the available commands, see the *CICS Operations and Utilities Guide*. For information about moving the contents of CICS tables to the CSD, see the *CICS Resource Definition Guide*.

**Note:** If you are using a CSD with an earlier release of CICS, upgrade your CSD as part of the process of migration. For information about upgrading the CSD, and about the release compatibility of the CSD after upgrading, see the *CICS Transaction Server for z/OS Migration from CICS TS Version 2.3*.

## Installing definitions for the Japanese language feature

If you have the Japanese language feature, install the definitions for the feature in the CSD, by running DFHCSDUP and specifying:

```
UPGRADE USING(DFHRDJPN)
```

For information about the DFHCSDUP utility program and the available commands, see the *CICS Operations and Utilities Guide*.

---

## CSD XRF considerations

If you are running CICS with XRF, both the active and alternate CICS regions must refer to the same CICS system definition data set (that is, the CSD must be passively shared). The active and alternate CICS regions can share the same CSD even if they are running on different MVS images. A CICS region running with XRF=YES may also share the CSD with other CICS regions within the same MVS image. (For a definition of passively and actively shared data sets in an XRF environment, see page “Actively and passively shared data sets” on page 29.)

The CSD is allocated by MVS when the CICS job step is initiated. This means the DD statements in the CICS startup job streams defining the CSD for the active and alternate CICS regions must specify DISP=SHR.

The alternate CICS region does not open the CSD during initialization, or before takeover occurs. The alternate CICS region does not even open the CSD during takeover, if the CSD was not changed at any time by the active CICS region. (For example, the CSD might have been used only to install a group list at CICS startup, and subsequently by read-only operations.) However, if you use the CEDA transaction in an active CICS region to alter resource definitions, the CSD might be opened at takeover, to perform any file backout that is necessary. To enable file backout to occur, you must define the CSD as a recoverable resource by the system initialization parameter CSDRECOV; see “Defining CSD attributes” on page 67.

For more information about using the CSD as a recoverable file, see “Planning for backup and recovery” on page 76.



---

## Chapter 8. Setting up and using catalog data sets

This section describes how to define and use the CICS **global** catalog data set (GCD), and the CICS **local** catalog data set (LCD), which CICS needs to catalog CICS system information. For the rest of this section, these data sets are referred to as the global catalog and the local catalog. (The CICS catalog data sets are not connected with MVS system catalogs, and contain data that is unique to CICS.)

### Notes:

1. You must define and initialize new CICS catalogs for CICS Transaction Server for z/OS, Version 3 Release 1.
2. If you redefine either one of the global and local catalogs, it is recommended that you redefine the other too.

For more information about how CICS uses the catalogs for startup and restart, see “The role of the CICS catalogs” on page 278.

---

### Defining the global catalog

The global catalog is a VSAM key-sequenced data set (KSDS). In an XRF environment there is only one global catalog. It is shared passively between the active and the alternate CICS regions. The global catalog is used:

- To record information that governs the possible types of start and the location of the CICS system log.
- During the running of CICS, to hold the resource definitions that are **installed** during initialization when CICS installs the group list, by the RDO CEDA INSTALL command or by the EXEC CICS CREATE command. These definitions can be for:
  - Files
  - Journals
  - Journalmodels
  - Mapsets
  - Programs
  - Pipelines
  - Sessions and connections, for communication with other CICS regions
  - Terminals, including any that are autoinstalled
  - Transactions
  - Transaction classes
  - Transient data queues
  - Web services
- During a normal (controlled) shutdown, to record terminal control information and profiles. (All other warm keypoint information is written to the CICS system log.)

For further information about what is written to the global catalog, and about how CICS uses the global catalog for startup and restart, see “The role of the CICS catalogs” on page 278.

## JCL to define and initialize the global catalog

Before its first use, you must define and initialize the CICS global catalog as a KSDS. You can use the sample job in Figure 13 to do this, or you can run the CICS-supplied job, DFHDEFDS, to define and initialize the global catalog as one of the data sets for the CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*.

```
//GLOCAT JOB accounting info,,CLASS=A
//DEFGCD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER -
        (NAME(CICSTS31.CICS.app1id.DFHGCD) - 1
        INDEXED -
        CYLINDERS(n1 n2) - 2
        FREESPACE(10 10) -
        SHAREOPTIONS(2) -
        RECORDSIZE(4089 4089)
        REUSE - 3
        VOLUMES(vol1id))
    DATA - 4
        (NAME(CICSTS31.CICS.app1id.DFHGCD.DATA) -
        CONTROLINTERVALSIZE(8192) - 5
        KEYS(28 0))
    INDEX -
        (NAME(CICSTS31.CICS.app1id.DFHGCD.INDEX) )
/*
//INITGCD EXEC PGM=DFHRMUTL,REGION=1M 6
//STEPLIB DD DSNAME=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD DD DSNAME=CICSTS31.CICS.app1id.DFHGCD,DISP=OLD
//SYSIN DD *
SET_AUTO_START=AUTOINIT
/* 7
```

Figure 13. Example job to define and initialize the global catalog

### Notes:

**1** The data set name in the CLUSTER definition must be the same as the DSN parameter in the DD statement for the global catalog in the CICS startup job stream.

**2** The primary and secondary extent sizes are shown as n1 and n2 cylinders. Calculate the size required to meet your installation's needs, and substitute your values for n1 and n2.

Whichever IDCAMS parameter you use for the GCD space allocation (CYLINDERS, TRACKS, or RECORDS), make sure that you specify a secondary extent. CICS abends if your GCD fills and VSAM cannot create a secondary extent.

**3** To enable the global catalog to be opened again and again as a reusable cluster, specify the REUSE option on the DEFINE CLUSTER command. REUSE should also be specified if you intend to use the COLD\_COPY input parameter of the DFHRMUTL utility.

**4** Specify a RECORDSIZE value for the global catalog. The default average and maximum record size is 4089 bytes. If your maximum record size is greater than 4089, you must specify this value instead in the sample job. For information about record sizes, see Table 14 on page 90.

**5** You can vary the CONTROLINTERVALSIZE from the values shown in the VSAM definition. However, although larger values reduce the number of control interval (CI) and control area (CA) splits, other factors increase CICS shutdown times, and slow down a cold start.

This job stream does not specify a BUFFERSPACE parameter, although you can code an explicit value if you want to define buffers of a specific size. BUFFERSPACE is the minimum bufferspace permitted; VSAM defaults to a bufferspace value equal to twice the CI size of the data component, plus the CI size of the index, which gives a default of 20480 bytes in the example job. A larger minimum buffer size (bufferspace) may improve cold start and warm restart times, and may significantly reduce CICS shutdown times.

Another way to define buffer space for the GCD is by means of the AMP parameter on the DD statement for the GCD in the CICS startup job stream, which you can use to override the default or defined value. (Note, however, that the BUFSP parameter defines the maximum bufferspace. If you define a BUFFERSPACE value on the AMP parameter that is smaller than the BUFFERSPACE value specified in the DEFINE statement, the BUFFERSPACE value takes precedence.

For performance reasons, CICS defines a STRNO (number of strings) value of 32. Based on the example job stream in Figure 13 on page 86, the absolute minimum value of BUFSP is calculated as follows:

```
#
#
#
#
BUFND = (STRNO + 1)
BUFNI = STRNO
BUFSP = 33 * 8192 (BUFND * CI size) + 32 * 1536 (BUFNI * CI size) =
        319488 bytes
```

**Note:** This is the smallest figure that can be used for BUFSP.

The principal factors affecting CICS startup and shutdown times are:

- The number of resources defined in the group list for those definitions managed by RDO
- The number of resources defined in CICS tables
- The size of the system log

**6** The job step INITGCD uses the recovery manager utility program, DFHRMUTL, to initialize the data set. DFHRMUTL writes a record to the data set, specifying that, on its next run using this global catalog, if START=AUTO is specified, CICS is to perform an initial start and not prompt the operator for confirmation. This record is called the autostart override record.

DFHRMUTL can also be used to override the type of start that would occur on an automatic startup, to be cold.

For full information about DFHRMUTL, and further examples of its use, see the *CICS Operations and Utilities Guide*.

In earlier releases of CICS, IDCAMS was used to write an initial record, using REPRO, to initialize the global catalog. Although you can still run this step, either

before or after running DFHRMUTL, this practice has been replaced by the use of DFHRMUTL to initialize the global catalog. See Figure 13 on page 86.

**7** It is recommended that you also run the DFHCCUTL utility in this same job. Run DFHRMUTL first and check its return code before running DFHCCUTL. If you do this, the global and local catalogs should never get out of step. For information about running DFHCCUTL, see the *CICS Operations and Utilities Guide*.

### Reusing the global catalog to perform a cold start

If you need to perform a cold start, **do not** delete and redefine the global catalog data set. If you were to delete and redefine the global catalog, CICS would perform an *initial* start, and all recovery information for remote systems would be lost. When remote systems reconnected, CICS would inform them that it had lost any information that they needed to resynchronize their units of work, and messages would be produced to record the fact, on both the local and the remote systems.

Instead, to specify that the next start should be cold, use the DFHRMUTL utility with the SET\_AUTO\_START=AUTOCOLD option. This has the following advantages:

- You do not have to reset the START system initialization parameter from AUTO to COLD, and back again.
- Because sufficient information is preserved on the global catalog and the system log, CICS is able to recover information for remote systems from the log, and to reply to remote systems in a way that enables them to resynchronize their units of work.

You can speed up a cold start by using the DFHRMUTL COLD\_COPY option to copy only those records that are needed for the cold start to another catalog data set. If the return code set by DFHRMUTL indicates that the copy was successful, a subsequent job-step can copy the new (largely empty) catalog back to the original catalog data set. The performance gain occurs because, at startup, CICS does not have to spend time deleting all the definitional records from the catalog. This technique will also speed up initial starts, for the same reason. Figure 14 on page 89 is an example of this technique.

**Note:** Before you use COLD\_COPY, you should be certain that you wish to perform a cold or initial start. As a safeguard, make a backup copy of the original global catalog before you copy the new catalog output by DFHRMUTL over it. For more information about the use of the global catalog in a cold start of CICS, see “Controlling start and restart” on page 278.

```

//RMUTL   EXEC PGM=DFHRMUTL,REGION=1M
//STEPLIB DD DSNAME=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD  DD DSNAME=CICSTS31.CICS.app1id.DFHGCD,DISP=OLD
//NEWGCD  DD DSNAME=CICSTS31.CICS.app1id.COPY.DFHGCD,DISP=OLD
//SYSIN   DD *
SET_AUTO_START=AUTOCOLD,COLD_COPY
/*
//      IF (RMUTL.RC<16) THEN
/* Steps to be performed if RMUTL was a success
//COPY   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//DFHGCD DD DSNAME=CICSTS31.CICS.app1id.DFHGCD,DISP=OLD
//NEWGCD DD DSNAME=CICSTS31.CICS.app1id.COPY.DFHGCD,DISP=OLD
//SYSIN   DD *
        REPRO INFILE(NEWGCD) OUTFILE(DFHGCD) REUSE
/*
/* End of steps to be performed if RMUTL was a success
//      ENDIF

```

Figure 14. DFHRMUTL example—setting the global catalog for a cold start. The *COLD\_COPY* option is used to improve startup performance. Note that the *NEWGCD* and *DFHGCD* data sets must have been defined with the *REUSE* attribute.

## Space calculations

Each global catalog record has a 28-byte key.

To estimate the amount of space needed in your global catalog to keypoint installed resource definitions, table entries, and control blocks, use the sizes specified in Table 14 on page 90.

Each entry is one VSAM record, and the records for each type of table have different keys.

The space requirements for a VSAM KSDS such as DFHGCD can vary for different CICS cold starts. This can occur even if no changes have been made to the CICS definitions to be stored on the VSAM KSDS. This is because VSAM will utilize the space in the data set differently depending on whether the data set has just initialized, or has data from a previous run of CICS. CICS will call VSAM to perform sequential writes. VSAM honors the 'freespace' value specified on the data set's definition if the keys of the records being added sequentially are higher than the highest existing key. However, if the data set contains existing records with a higher key than the ones being inserted, 'freespace' is only honored once a CI split has occurred.

The size of the index portion of the data set may also vary depending on the number of CI and CA splits that have occurred. This affects the index sequence set.

When you are initializing the global catalog, you can use the *COLD\_COPY* parameter, (*SET\_AUTO\_START=AUTOCOLD,COLD\_COPY*). The cold copy will create a reduced copy of the global catalog data set, that will improve the performance of the cold start. The CI splits will cease after the first cold start, and the data set will not expand into additional extents. Another solution is to reorganize or reinitialize the data set from time to time.

Table 14. Sizes for entries in the global catalog

<b>Installed definition, table entry, or control block</b>	<b>Number of bytes per entry</b>
Installed PARTNER definitions	96 bytes
Installed program definitions	44 bytes
Installed indirect queue definition	92 bytes
Installed intrapartition queues definition	236 bytes
Installed extrapartition queue definition	296 bytes
Installed remote queue definition	84 bytes
Installed TRANSACTION definitions (without TPNAME)	112 bytes
Installed TRANSACTION definitions (with TPNAME or XTPNAME)	176 bytes
Installed VSAM file (or data table) definition	260 bytes
Installed TRANCLASS definitions	8 bytes
BDAM file control table entry (FCT)	118 bytes
BDAM data control blocks	112 bytes
VSAM LSR share control blocks <b>1</b>	1156 bytes
Data set names (JCL or dynamically allocated) <b>2</b>	52 bytes
Data set name blocks	115 bytes
File control recovery blocks <b>3</b>	97 bytes
Terminal control table entry (TCT)	1500 bytes (approx)
Dump table entry	48 bytes
Interval control element (ICE)	68 bytes
Automatic initiator descriptor (AID)	68 bytes
Transient data destination record	18 bytes
Transient data destination auxiliary record	6 bytes
Installed TYPETERM definitions <b>4</b>	582 bytes
Installed model TERMINAL definitions <b>4</b>	582 bytes
Deferred work element (DWE) <b>5</b>	80 bytes
Installed journal	88 bytes
Installed journalmodel	80 bytes
Recovery manager remote names	106 bytes
Installed PIPELINE definition	1384 bytes
Installed TCPIP SERVICE definition	484 bytes
Installed WEBSERVICE definition	900 bytes
Installed DOCTEMPLATE definition	128 bytes

#  
#  
#  
#

**Notes:**

**1** One for each LSR pool, i.e. 8.

**2** If you open a VSAM path you get two of these, for BDAM or VSAM base data sets you get one.

**3** You will only have these if you use the VSAM RLS SHCDS option NONRLSUPDATEPERMITTED. In this case, for each data set that you have specified NONRLSUPDATEPERMITTED for, you could have an upper limit. This limit is the number of different file names through which you access the data set multiplied by the number of tasks that update the data set. You will normally only have a few, if any, of these control blocks.

**4** The TYPETERM and model TERMINAL definitions are present if you are using autoinstall. They are stored directly in the global catalog when the definitions are installed, either by a CEDA transaction, or as members of a group installed through

a group list. For example, if you start up CICS with the startup parameter GRPLIST=DFHLIST, the CICS-supplied TYPETERM and model terminal definitions, defined in the groups DFHTERM and DFHTYPE, are recorded in the global catalog. Allow space in your calculations for all autoinstall resources installed in your CICS region.

**5** The value given is for a DWE chained off an LU6.1 session, or an APPC session.

## Job control statement for CICS execution

If you define the global catalog using the sample job in Figure 13 on page 86, the data definition statement for the CICS execution is:

```
//DFHGCD DD DSN=CICSTS31.CICS.app1id.DFHGCD,DISP=OLD
```

This is a minimum specification for a global catalog for use by a single CICS region. Add the relevant AMP subparameters to help improve restart and shutdown time. The AMP parameter is described in the *OS/390 MVS JCL Reference* manual, and an example is shown in the CICS startup job stream in Chapter 21, “CICS startup,” on page 337.

If you are running CICS with XRF, the global catalog is passively shared by the active and alternate CICS regions, and you must specify DISP=SHR.

---

## Defining the local catalog

CICS Transaction Server for z/OS is divided into functional areas (or components) known as domains. These domains communicate through a central component, the CICS kernel, and their initialization and termination is controlled by the domain manager. The kernel, the domain manager, and the other domains all require an individual domain parameter record, and these are stored in the local catalog.

The CICS domains use the local catalog to save some of their information between CICS runs, and to preserve this information across a cold start. For further guidance information about what is written to the local catalog, and about how CICS uses the local catalog for startup and restart, see “Controlling start and restart” on page 278.

The local catalog is a VSAM key-sequenced data set (KSDS). It is not shared by any other CICS region, such as an alternate CICS in an XRF environment. If you are running CICS with XRF, you must define a unique local catalog for the active CICS region, and another for the alternate CICS region.

Unlike the global catalog, which must be defined with enough space to cope with any increase in installed resource definitions, the size of the local catalog is relatively static. The following section describes the information held on the local catalog.

## Information written to the local catalog

Before the local catalog can be used to bring up a CICS region, it must be initialized with the following data:

- The domain manager parameter records, each of which contains information relating to one of the CICS domains. These records are identified by their domain names, which are:

Domain name in catalog	Description
------------------------	-------------

<b>DFHAP</b>	Application domain.
<b>DFHBA</b>	Business application manager.
<b>DFHCC</b>	CICS local catalog domain.
<b>DFHDD</b>	Directory manager domain.
<b>DFHDDH</b>	Document handler domain.
<b>DFHDM</b>	Domain manager domain.
<b>DFHDS</b>	Dispatcher domain
<b>DFHDU</b>	Dump domain.
<b>DFHEM</b>	Event manager domain
<b>DFHGC</b>	CICS global catalog domain.
<b>DFHKE</b>	Kernel domain.
<b>DFHLD</b>	Loader domain.
<b>DFHLG</b>	Log manager domain.
<b>DFHLM</b>	Lock manager domain.
<b>DFHME</b>	Message domain.
<b>DFHMN</b>	Monitoring domain.
<b>DFHNQ</b>	Enqueue manager domain
<b>DFHPA</b>	System initialization parameter domain.
<b>DFHPG</b>	Program manager domain.
<b>DFHRM</b>	Recovery manager domain
<b>DFHRX</b>	RRMS domain
<b>DFHSH</b>	Scheduler services domain
<b>DFHSM</b>	Storage manager domain.
<b>DFHSO</b>	Sockets domain.
<b>DFHST</b>	Statistics domain.
<b>DFHTI</b>	Timer domain.
<b>DFHTR</b>	Trace domain.
<b>DFHUS</b>	User domain.
<b>DFHWB</b>	Web domain
<b>DFHXM</b>	Transaction manager domain.
<b>DFHXS</b>	Security domain.

- Four loader domain parameter records, which contain information relating to:
  - DFHDMP, the CSD file manager
  - DFHEITSP, the RDO language definition table
  - DFHPUP, the CSD parameter utility program

To enable you to initialize the local catalog correctly, with all the records in the correct sequence, there is a CICS-supplied utility called DFHCCUTL that you run immediately after you have defined the VSAM data set.

In addition to the information written to the local catalog when you first initialize it, the loader domain writes a program definition record for each CICS nucleus module. The number of records varies depending on the level of function you have included in your CICS region.

Allow for at least 75 of these loader-domain records.

Some domains also write a domain status record to the local catalog, for use in a warm or emergency restart. For example, in the case of the dump domain, the status record indicates which transaction dump data set was in use during the previous run. The domains that write to the local catalog are:

- Dispatcher domain
- Dump domain
- Loader domain
- Message domain
- Parameter manager domain
- Storage manager domain
- Transient data

You can add records to the local catalog to enable the CICS self-tuning mechanism for storage manager domain subpools. For details of how to do this using the CICS-supplied utility program, DFHSMUTL, see the *CICS Operations and Utilities Guide*.

Finally, when you define the VSAM cluster for the local catalog, specify a secondary extent value as a contingency allowance. See the sample job in Figure 15 on page 94.

## **Job control statements to define and initialize the local catalog**

Before its first use, you must define and initialize the CICS local catalog as a VSAM key sequenced data set (KSDS). To do this, you can use the sample job in Figure 15 on page 94. Alternatively, you can run the CICS-supplied job DFHDEFDS to create a local catalog for an active CICS region or the CICS-supplied job DFHALTDS to create a local catalog for an alternate CICS region. For information about the jobs DFHDEFDS and DFHALTDS, see the *CICS Transaction Server for z/OS Installation Guide*.

```

//LOCAT    JOB accounting info,,CLASS=A
//DEFLCD   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
/*
      DEFINE CLUSTER -
          (NAME( CICSTS31.CICS.applid.DFHLCD) -           1
           INDEXED -
           RECORDS( 200 10 ) -                           2
           FREESPACE(10 10) -
           SHAREOPTIONS( 2 ) -
           REUSE -
           VOLUMES( volid ))
      DATA
          (NAME( CICSTS31.CICS.applid.DFHLCD.DATA ) -
           KEYS( 28 0 ) -
           RECORDSIZE( 45 124 ) -                       3
           CONTROLINTERVALSIZE( 2048 ))
      INDEX (NAME( CICSTS31.CICS.applid.DFHLCD.INDEX ) )
/*
//*****
//INITLCD EXEC PGM=DFHCCUTL
/*
//*          INITIALIZE THE CICS LOCAL CATALOG
/*
//STEPLIB DD DSN=CICSTS31.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DFHLCD  DD DSN=CICSTS31.CICS.applid.DFHLCD,DISP=SHR
/*
//

```

Figure 15. Sample job to define and initialize the local catalog

**Notes:**

**1** If you are defining local catalogs for multiple CICS regions (for example, for active and alternate CICS regions when running with XRF), you can identify the clusters uniquely by making the specific APPLID of each CICS one of the data set qualifiers. For example, you could use the following names for the clusters of active and alternate CICS regions, where DBDCCIC1 and DBDCCIC2 are the specific APPLIDs:

```

DEFINE CLUSTER -
    (NAME( CICSTS31.CICS.DBDCCIC1.DFHLCD)
    :
    :
DEFINE CLUSTER -
    (NAME( CICSTS31.CICS.DBDCCIC2.DFHLCD)
    :
    :

```

**2** Space for about 200 records should be adequate for the local catalog, but also specify space for secondary extents as a contingency allowance.

**3** The local catalog records are small by comparison with the global catalog. Use the record sizes shown, which, in conjunction with the number of records specified, ensure enough space for the data set.

**Job control statement for CICS execution**

If you define the local catalog using the sample job in Figure 15, the data definition statement for the CICS execution is:

```

//DFHLCD DD DSN=CICSTS31.CICS.applid.DFHLCD,DISP=OLD

```

---

## Chapter 9. Setting up and using auxiliary trace data sets

This section describes the auxiliary trace data sets controlled by CICS.

Several types of tracing are available in CICS to help you with problem determination, and these are described in the *CICS Problem Determination Guide*. Among the various types of trace, the CICS tracing handled by the CICS trace domain allows you to control the amount of tracing that is done, and also to choose from any of three destinations for the trace data. Any combination of these three destinations can be active at any time:

1. The internal trace table, in main storage above the 16MB line in the CICS address space.
2. The auxiliary trace data sets, defined as BSAM data sets on disk or tape.
3. The MVS generalized trace facility (GTF) data sets.

For information about GTF, see the *OS/390 MVS Diagnosis: Tools and Service Aids* manual. For information about using CICS tracing for problem determination, see the *CICS Problem Determination Guide*.

If you decide to use auxiliary trace, you must define one or two sequential data sets, on either disk or tape. If you specify automatic switching for your auxiliary trace data sets, define two data sets. If you specify autoswitch for auxiliary trace, and define only one data set, auxiliary trace is stopped and CICS takes a dump.

The DD names of the auxiliary trace data sets are defined by CICS as DFHAUXT and DFHBUXT. If you define a single data set only, its DD name must be DFHAUXT. You can allocate and catalog the auxiliary trace data sets before starting CICS.

If you use tape for recording auxiliary trace output, use unlabeled tape. Using standard-labeled tape, whether on a single tape drive or on two tape drives, stops you processing the contents of any of the volumes with the DFHTU640 utility until after the CICS step has been completed. If you use standard-labeled tape, make sure all the output produced in the CICS run fits on the one (or two) volumes mounted.

You cannot catalog data sets that are on unlabeled tapes.

---

### Starting and controlling auxiliary trace

You may need to use auxiliary trace data sets to avoid the loss of diagnostic information, because internal trace table entries wrap around. When the end of the internal trace table is reached, subsequent entries overwrite those at the start of the table. To get a trace of CICS activity in which trace entries are not overwritten, use the auxiliary trace data sets. This can be particularly useful if you are using CICS trace during startup, because of the high volume of trace entries written when CICS is initializing.

You can start CICS tracing at initialization by coding system initialization parameters; you can also specify which of the three destinations you want CICS to use. The following is a summary of the trace system initialization parameters that you can code:

Keyword	Description
<b>AUXTR</b>	Switches auxiliary trace on or off at CICS startup.
<b>AUXTRSW</b>	Specifies automatic switching for auxiliary trace data sets when full.
<b>INTTR</b>	Switches internal trace on or off at CICS startup.
<b>GTFTR</b>	Specifies whether CICS is to use GTF as a destination for CICS trace data.
<b>SPCTR</b>	Specifies the level of special tracing.
<b>SPCTRxx</b>	Specifies the level of special tracing for the “xx” component.
<b>STNTR</b>	Specifies the level of CICS standard tracing.
<b>STNTRxx</b>	Specifies the level of standard tracing for the “xx” component.
<b>SYSTR</b>	Switches the system master trace flag on or off at CICS startup.
<b>TRTABSZ</b>	Defines the size of the CICS internal trace table.
<b>USERTR</b>	Switches the user trace flag on or off at CICS startup.

For more information about these system initialization parameters, and how to code them, see Chapter 18, “Specifying CICS system initialization parameters,” on page 151.

You can also control CICS tracing by means of the CICS-supplied transactions CETR and CEMT. (Note that you cannot use CETR through an MVS console.) For guidance information about the CICS control options available with CETR and CEMT, see *CICS Supplied Transactions*.

---

## Allocating auxiliary trace data sets

If you are defining auxiliary trace data sets on disk, you can use the job shown in Figure 16 to allocate and catalog them before running CICS.

Alternatively, you can run the CICS-supplied job DFHDEFDS to create the auxiliary trace data sets for an active CICS region or the CICS-supplied job DFHALTDS to create them for an alternate CICS region. For information about the jobs DFHDEFDS and DFHALTDS, see the *CICS Transaction Server for z/OS Installation Guide*.

```

//DEFTRCDS JOB (accounting information),
//          MSGCLASS=A,MSGLEVEL=(1,1),
//          CLASS=A,NOTIFY=userid
//*****
//*          Create auxiliary trace data sets
//*****
//ALLOCD   EXEC PGM=IEFBR14
//DFHAUXT DD DSN=CICSTS31.CICS.applid.DFHAUXT,UNIT=3380,VOL=SER=vol1d,
//          DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),
//          SPACE=(CYL,(5,1))
//DFHBUXT DD DSN=CICSTS31.CICS.applid.DFHBUXT,UNIT=3380,VOL=SER=vol1d,
//          DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),
//          SPACE=(CYL,(5,1))
//

```

Figure 16. Sample job to define auxiliary trace data sets on disk

## Notes:

**1** The DCB subparameters shown in this sample job specify the required DCB attributes for the CICS auxiliary trace data sets. As an alternative to this job, you can specify (NEW,CATLG) on the DD statements in the CICS startup job stream, omit the DCB parameter, and let CICS open the data sets with the same default values.

**2** Change the space allocations in this sample job stream to suit your installation's needs.

## Space calculations

Trace entries are of variable length, but the physical record length (block size) of the data written to the auxiliary trace data sets is fixed at 4096 bytes. As a rough guide, each block contains an average of 40 entries, although the actual number of entries depends on the processing being performed.

## Job control statements for CICS execution

If you allocate and catalog the auxiliary trace data sets on disk as shown in Figure 16 on page 96, you can define them to CICS in the startup job stream with the following DD statements:

```
//DFHAUXT DD DSN=CICSTS31.CICS.appIid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
//DFHBUXT DD DSN=CICSTS31.CICS.appIid.DFHBUXT,DCB=BUFNO=n,DISP=SHR
```

If you specify BUFNO greater than 1, you can reduce the I/O overhead involved in writing auxiliary trace records. A value between 4 and 10 can greatly reduce the I/O overhead when running with auxiliary trace on.

DISP=SHR allows the simultaneous processing of a data set by the DFHTU640 offline utility program after a switch to the other data set has taken place.

For auxiliary trace data sets on unlabeled tapes, use the following sample DD statements:

```
//DFHAUXT DD DSN=CICSTS31.CICS.appIid.DFHAUXT,UNIT=3400,VOL=SER=valid,
//          DISP=(NEW,KEEP),LABEL=(,NL)
//DFHBUXT DD DSN=CICSTS31.CICS.appIid.DFHBUXT,UNIT=3400,VOL=SER=valid,
//          DISP=(NEW,KEEP),LABEL=(,NL)
```

If you are using tape for the auxiliary data sets, assign tape units and mount the tapes before entering the command to start auxiliary trace. If you specify AUXTR=ON as a system initialization parameter, ensure the tape is mounted before starting CICS.

## XRF considerations

The active and the alternate CICS regions must refer to different auxiliary trace data sets; that is, they must be unique data sets. This means that you can capture auxiliary trace data for the active CICS region, while the alternate CICS region is running but before takeover occurs.

For the active CICS region, you use CETR or CEMT to control auxiliary trace data sets. For the alternate CICS region, you use CEBT. For information about using these transactions, see *CICS Supplied Transactions*.

---

## Using the trace utility program (DFHTU640)

If you write trace entries to CICS auxiliary trace data sets you can use the trace utility program, DFHTU640, to extract all or selected trace entries, and format and print the data.

To process the separate trace data sets for active and alternate CICS regions, you need separate utility jobs for each set of data sets. For information about DFHTU640, see the *CICS Operations and Utilities Guide*.

---

## Chapter 10. Defining dump data sets

This chapter describes how to define the following two types of dump data sets that CICS uses for recording dumps as a consequence of a failure detected during CICS execution, or upon explicit request:

1. CICS transaction dump data sets, for recording transaction dumps.
2. MVS system dump data sets, for recording system dumps that CICS requests using the MVS SDUMP macro.

CICS has a dump table facility that enables you to control dumps. The dump table lets you:

- Specify the type of dump, or dumps, you want CICS to record.
- Suppress dumping entirely.
- Specify the maximum number of dumps to be taken during a CICS run.
- Control whether CICS is to terminate as a result of a failure that results in a dump.

You can set the options you want in the dump table in two ways:

1. Using the CEMT master terminal command
2. Using the EXEC API commands

When you start CICS for the first time, CICS uses system default values for the dump table options, and continues to use the system default values until you modify them with a CEMT or EXEC CICS command. For information about the dump table options you can set, see the *CICS Problem Determination Guide*.

**Note:** The MVS system dump data sets can become full with unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after message DFHAP0001 or DFHSR0001). To prevent this from happening, you can suppress all SDUMPs preceding ASRA, ASRB and ASRD abends, or you can suppress some of them. “Suppressing system dumps that precede ASRx abends” on page 100 tells you how to do this.

---

### System dumps

CICS produces a system dump using the MVS SDUMP macro.

### MVS SDUMP macro

The MVS SDUMP dump results from CICS issuing an MVS SDUMP macro. It includes almost the entire CICS address space, that is, the MVS nucleus and other common areas, as well as the CICS private storage areas. The SDUMP dump is written to an MVS dump data set, which you can process using the interactive problem control system (IPCS). For information about the SDUMP macro, and the associated MVS dump data sets, see the *OS/390 MVS Diagnosis: Procedures manual*.

The SDUMP macros issued by CICS normally contain the QUIESCE=NO parameter. They may not if the SDUMP is taken because of an abend in CICS SVC code or when altering MRO control blocks. This parameter allows the MVS system to remain dispatchable while the SDUMP is being taken, thus reducing the impact on the system. However if QUIESCE=YES is specified as an MVS system default it

will override that specified by CICS. These defaults can be altered by using the MVS CHNGDUMP command. For more information on this command see the *OS/390 MVS System Commands* manual.

You should use the MERGE function when changing the SDUMP options by the CHNGDUMP command to ensure that the areas selected by CICS to dump are included in the MVS dump data set output. If you use the ADD option, it replaces any options specified by CICS when issuing the SDUMP in many cases. This can result in partial dumps being taken to the MVS dump data set. MVS always includes LSQA and TRT in the dump but may exclude the private area if you use the wrong options in the update by the CHNGDUMP command. You must thoroughly review your use of the CHNGDUMP command when setting up your CICS region. For information about the CHNGDUMP command and the effect that altering its options has on the dump output from CICS, see the *OS/390 MVS Initialization and Tuning Guide*.

If you are running CICS with XRF, the surveillance signal of the active CICS region stops during an MVS SDUMP of the active CICS region's address space, which could lead to unnecessary takeovers being initiated, if the ADI (alternate delay interval) for the alternate is set too low. However, you can prevent SDUMPs of other address spaces from causing unnecessary takeovers when the alternate CICS is running on a different MVS image by setting the QUIESCE=NO option for SDUMP, using the MVS CHNGDUMP command.

## Suppressing system dumps that precede ASRx abends

The MVS system dump data sets can become full with unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after either message DFHAP0001 or DFHSR0001).

If CICS storage protection is active, you can suppress the system dumps caused by errors in application programs (after message DFHSR0001), while retaining the dumps caused by errors in CICS code (after message DFHAP0001). To do this, use either a CEMT SET SYDUMPCODE command, or an EXEC CICS SET SYSDUMPCODE command to suppress system dumps for system dumpcode SR0001.

```
CEMT SET SYDUMPCODE(SR0001) ADD NOSYSDUMP
```

CICS uses dumpcode SR0001 if an application program was executing in user-key at the time of the program check or MVS abend. This is only possible if storage protection is active. If the program was executing in CICS-key, dumpcode AP0001 is used instead.

Where storage protection is not active, SDUMPs can be suppressed by suppressing dumpcode AP0001. However, note that this suppresses dumps for errors in both application *and* CICS programs. The XDUREQ Global User Exit can be used to distinguish between AP0001 situations in application and CICS programs.

For more information about the storage protection facilities available in CICS, see "Storage protection" on page 354.

If you want SDUMPs for one of these transaction abends but not the other, select the one you want by using either a CEMT TRDUMPCODE or an EXEC CICS TRANDUMPCODE command. This specifies, on an entry in the dump table, that SDUMPs are to be taken for either ASRA, ASRB, or ASRD abends. 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, in this case the SDUMPs are taken at a later point than SDUMPs usually taken for system dump code AP0001 and SR0001.

For information about the DFHAP0001 and DFHSR0001 messages, see the *CICS Messages and Codes* and the *CICS Problem Determination Guide*.

## Processing system dumps

You can process a system dump using IPCS, either online under TSO, or by submitting a batch job to print it. IPCS is described in the *OS/390 MVS IPCS User's Guide*, GC28-1756. For information about the IPCS VERBEXIT parameters that you use with the CICS IPCS dump exit, see the *CICS Operations and Utilities Guide*.

---

## The CICS transaction dump data sets

CICS records transaction dumps on a sequential data set, or a pair of sequential data sets, on disk or tape. The data sets must be defined in the CICS run with the DD names DFHDMPA and DFHDMPB, but if you define a single data set only, its DD name must be DFHDMPA. In this chapter, a reference to “the CICS dump data set” means either DFHDMPA or DFHDMPB. Note that CICS always attempts to open at least one transaction dump data set during initialization. If you do not include a DD statement for at least one transaction dump data set in your CICS job, initialization continues after the following message is sent to the console:

```
DFHDU0306 applid Unable to open Transaction Dump Data set
          dataset-text-descr
```

With two data sets, you can print transaction dumps from one data set while CICS is running. To do this, first use CEMT SET DUMP SWITCH to switch the data sets. CICS closes the current data set after any transaction dump being recorded has been completed, and opens the other data set. You can print the completed data set with the DFHDU640 dump utility program. For information about the DFHDU640 dump utility program, see the *CICS Operations and Utilities Guide*.

In addition to switching dump data sets explicitly, the operator can use CEMT SET DUMP AUTO to cause automatic switching when the current data set becomes full. (Note that this permits **one** switch only.) When a transaction dump data set is full, CICS closes the data set and issues console messages as follows:

```
DFHDU0303I applid Transaction Dump Data set dataset closed.
DFHDU0304I applid Transaction Dump Data set dataset opened.
DFHDU0305I applid Transaction Dump Data set switched to ddname.
```

where “x” and “y” can have the value A or B. If you specified DISP=SHR for the dump data set, you can print the completed data set with the DFHDU640 utility program and then reissue the command: CEMT SET DUMP AUTO. This again switches data sets automatically (once only) when the current data set is full.

You can define the CICS dump data sets DFHDMPA and DFHDMPB as temporary data sets for each CICS run. More commonly, you allocate and catalog them in advance, reuse them repeatedly, and do not delete them when the CICS job has completed. You can then use the DFHDU640 utility program to print the dump output at any time during or after the CICS run. Note that it is not practical to try to use temporary data sets when running CICS with XRF.

You do not need DCB parameters for dump data sets (but see “Copying disk dump data sets to tape” on page 103 for an exception). When CICS opens the dump data set, it issues an MVS DEVTYPE macro. This returns the track size for direct access devices, or 32760 for magnetic tape. The maximum block size used for a transaction dump is the lesser of the values returned from the DEVTYPE macro and 4096. As this usually results in a block size of 4096 (because devices generally have a track size greater than this), CICS writes multiple blocks per track. After writing each block, MVS returns the amount of space remaining on the current track. If the space remaining is 256 bytes or more, then the size of the next block written is the lesser of the values returned by MVS and 4096.

If the space remaining is less than 256 bytes, the next block is written to the next track.

There are four global user exits that you can use with the transaction dump data sets:

1. XDUCLE, after the dump domain has closed a transaction dump data set
2. XDUREQ, before the dump domain takes a transaction dump
3. XDUREQC, after the dump domain takes a transaction dump
4. XDUOUT, before the dump domain writes a record to the transaction dump data set

For programming information about the global user exits, see the *CICS Customization Guide*

## Selecting the transaction dump data set at startup

You can code the DUMPDS system initialization parameter to specify which transaction dump data set is to be opened during CICS initialization. If you specify DUMPDS=AUTO, CICS opens, on a warm or emergency start, the data set that was **not** in use when CICS was last terminated. This lets you restart CICS after an abnormal termination without waiting to print the dump data set that was in use at termination. For more information about the DUMPDS parameter, see Chapter 18, “Specifying CICS system initialization parameters,” on page 151.

---

## Job control statements to allocate dump data sets

You can run the CICS-supplied job DFHDEFDS to allocate and catalog the dump data sets for an active CICS region or the CICS-supplied job DFHALTDS to allocate and catalog them for an alternate CICS region. For information about the jobs DFHDEFDS and DFHALTDS, see the *CICS Transaction Server for z/OS Installation Guide*.

Alternatively, you can use the sample data definition statements in Figure 17 to allocate and catalog dump data sets on disk.

```
//DFHDMPA DD DSN=CICSTS31.CICS.app1id.DFHDMPA,DISP=(NEW,CATLG),
//          UNIT=3380,VOL=SER=vol1id,SPACE=(CYL,(5,1))
//DFHDMPB DD DSN=CICSTS31.CICS.app1id.DFHDMPB,DISP=(NEW,CATLG),
//          UNIT=3380,VOL=SER=vol1id,SPACE=(CYL,(5,1))
```

Figure 17. Sample job control statements for defining disk dump data sets

**Note:** Change the space allocations in this sample job stream to suit your own installation’s needs.

If you are running CICS with XRF, you must allocate different data sets for the alternate.

If you use tape for recording dump output, use unlabeled tape. Standard-labeled tape, whether on a single tape drive or on two tape drives, stops you processing the contents of any of the volumes with the DFHDU640 utility until after the CICS step has been completed. If you want to use standard-labeled tape, make sure that all the output produced in the CICS run fits on the one or two volumes mounted.

You cannot catalog dump data sets defined on unlabeled tapes. Your data set definitions must be in the CICS startup job stream each time CICS is run.

## Copying disk dump data sets to tape

If you intend copying dump data sets to tape or disk, you must specify DCB parameters on the DD statements when allocating and cataloging the dump data sets, as follows:

```
//          DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092)
```

Otherwise, if you do not intend copying dump data sets to tape or disk, it is not necessary to include DCB parameters when defining dump data sets on disk, as illustrated in the sample job in Figure 17 on page 102.

## Space calculations

For the initial installation of CICS, a dump data set of between 5 and 10MB should be enough. When normal operation begins, you can adjust this to suit your own installation's requirements.

---

## Job control statements for CICS execution

The following DD statements for inclusion in the CICS startup job stream assume that the transaction dump data sets have been cataloged previously:

```
//DFHMPA DD DSN=CICSTS31.CICS.app1id.DFHMPA,DISP=SHR
//DFHMFB DD DSN=CICSTS31.CICS.app1id.DFHMPB,DISP=SHR
```

DISP=SHR enables each data set, if held on disk, to be processed by the DFHDU640 offline utility after the switch to the other data set has taken place.

The following are examples of DD statements for transaction dump data sets on unlabeled tapes:

```
//DFHMPA DD DSN=CICSTS31.CICS.app1id.DFHMPA,UNIT=3400,VOL=SER=vo1id1,
//          DISP=(NEW,KEEP),LABEL=(,NL)
//DFHMFB DD DSN=CICSTS31.CICS.app1id.DFHMPB,UNIT=3400,VOL=SER=vo1id2,
//          DISP=(NEW,KEEP),LABEL=(,NL)
```



---

## Chapter 11. Defining the CICS availability manager data sets

This chapter tells you how to define the CICS availability manager (CAVM) data sets. The CAVM is the mechanism that enables active and alternate CICS regions to coordinate their processing when XRF=YES is coded as a system initialization parameter. If you code XRF=NO, these data sets are not used. The CAVM requires two data sets: the XRF control data set and the XRF message data set.

This pair of data sets is logically a single entity that contains:

- State data whose main purpose is to ensure that, at any given time, only one job is allowed to fulfill the active role for a particular generic APPLID.
- Primary and secondary surveillance signals of active and alternate CICS regions, so that each CICS region can tell whether its partner is working correctly.
- Messages about the state of particular resources in use on the active CICS region, that are written by the active CICS region, and read and processed by the alternate CICS region.

Both the active and alternate CICS regions must refer to the same pair of data sets. You define these data sets, but must not try to initialize them, and you are recommended to place the data sets on separate volumes. The first time they are used, CICS recognizes them as a new pair of data sets. If they are new, CICS initializes them in such a way that, from then on, they can be used only as a pair with the original generic APPLID and for their original purpose (that is, as either an XRF message data set or an XRF control data set). If you need to redefine either data set, for any reason, you must redefine both of them.

You must define a separate pair of data sets for each generic APPLID in use. If a CICS complex consists of, for example, five regions, five pairs of data sets must be defined.

You do not need to take backup copies of these data sets because when neither of the active or alternate CICS regions is running, you can always start with a fresh pair of data sets.

**Why have two data sets?** Because of RESERVE commands issued by other MVS images in a multi-MVS environment, a shared DASD volume may become inaccessible for periods ranging from milliseconds to perhaps a minute. By making use of two data sets, placed on different volumes, CAVM can greatly reduce the risk that, by preventing surveillance signals from being written, normal RESERVE activity might cause the unnecessary takeover of a CICS region that was running normally.

If the access paths to the two volumes are separate, CAVM is also less vulnerable to hardware failures.

---

### The XRF control data set

The XRF control data set is used:

- To record the presence or absence, identities, and current states of the active and alternate CICS regions' jobs
- For the primary surveillance signals of the active and the alternate CICS regions

CAVM rejects a request from a CICS job to sign on as the active CICS region if the XRF control data set shows that an active CICS region is already present, or that a takeover is in progress. This ensures that the integrity of files and databases cannot be lost as a result of uncontrolled concurrent updating by two or more active CICS regions. As soon as an active or alternate CICS regions signs on, it starts to write its own surveillance signals, and to look for its partner's surveillance signals.

## JCL to define the XRF control data set

You must define the XRF control data set, but not initialize it. You can use the JCL statements in Figure 18 to define the XRF control data set. Alternatively, you can run the CICS-supplied job DFHDEFDS to define the XRF control data set as one of the data sets for a CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*>.

```

//CICSCTL JOB 'accounting info',name,MSGCLASS=A
//XRCTL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
        DEFINE CLUSTER -
            (NAME(CICSTS31.CICS.applid.DFHXRCTL) -
             RECORDSIZE(4089 4089) -
             CONTROLINTERVALSIZE(4096) -
             RECORDS(4) -
             NIXD -
             SHAREOPTIONS(3,3) -
             VOLUMES(vol1id1) -
             DATA -
             (NAME(CICSTS31.CICS.applid.DFHXRCTL.DATA))
/*
//

```

Figure 18. Sample job to define the XRF control data set

### Notes:

- 1** The RECORDSIZE must be at least 4089.
- 2** The control interval sizes of the XRF control data set and the XRF message data set must be equal, and at least 4096 bytes.
- 3** The SHAREOPTIONS must be specified as 3,3.
- 4** The data set must be VSAM-ESDS.

### Serializing access to the XRF control data set

Access to the XRF control data set must be serialized during the critical sections of CAVM signon, sign-off, and takeover processing. The correct choice of volume is important because this serialization is provided by RESERVE/RELEASE (DEQ) logic. For example, it would be unwise to place an XRF control data set on the same volume as the JES checkpoint data set. If you use global resource serialization (GRS) you must not convert this RESERVE, which uses the qname SYSCICSX, to a global ENQ.

## Space calculations

Only four control intervals are needed.

## Job control statements for CICS execution

The DD name required for CICS execution is DFHXRCTL. The following is an example of the JCL statement required:

```
//DFHXRCTL DD DSN=CICSTS31.CICS.app1id.DFHXRCTL,DISP=SHR
```

---

## The XRF message data set

The XRF message data set is used:

- Mainly to pass messages about the current states of specific resources from the active to the alternate CICS region.
- For the secondary surveillance signals of the active and alternate CICS regions, when the control data set is unavailable for this purpose, either because the last write has not completed yet or because of I/O errors.

## JCL to define the XRF message data set

Like the XRF control data set, the XRF message data set must be defined but not initialized. You can use the sample job in Figure 19 to define the XRF message data set. Alternatively, you can run the CICS-supplied job DFHDEFDS to define the XRF message data set as one of the data sets for a CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*.

If you use the sample JCL in Figure 19, read the accompanying notes; the other options shown are suggestions only.

You should define the XRF message data set on a volume that is **not** subject to RESERVE activity, and should not locate it where a single failure can make both it and the XRF control data set inaccessible. This reduces the risk of the surveillance signal being stopped accidentally while CICS is still running normally.

The XRF message data set is reserved for a short time for formatting when CICS uses it for the first time.

```
//CICSMMSG JOB 'accounting info',name,MSGCLASS=A
//XRMSG EXEC PGM=IDCAMS
//DDNAME2 DD DISP=OLD,UNIT=3380,VOL=SER=vo1id2
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
        DEFINE CLUSTER -
            (NAME(CICSTS31.CICS.app1id.DFHXRMMSG) -
             RECORDSIZE(4089 4089) -
             CONTROLINTERVALSIZE(4096) -
             RECORDS(1500) -
             NIXD -
             SHAREOPTIONS(3,3) -
             VOL(vo1id2) -
             FILE(DDNAME2)) -
            DATA -
            (NAME(CICSTS31.CICS.app1id.DFHXRMMSG.DATA))
/*
//
```

Figure 19. Sample job to define the XRF message data set

## Notes:

- 1** The RECORDSIZE must be at least 4089.
- 2** The control interval sizes of the XRF message data set and the XRF control data set must be equal, and at least 4096 bytes.

If the CI size of the XRF message data set is greater than 4096, the CI buffers occupy more real storage and virtual storage above the 16MB line, although fewer I/O operations occur during the “catch-up” phase.

- 3** The SHAREOPTIONS must be specified as 3,3.
- 4** The data set must not be indexed.

## Space calculations

It is difficult to give a simple answer to the question: “How big should my XRF message data set be?” A simple answer is that the size required depends on the length and number of messages that have been sent by the active CICS region but not yet received by the alternate CICS region.

The XRF message data set is written and read cyclically. When the alternate CICS region has read a message, that space becomes available for another message on the next cycle. It is important to make the data set large enough to store the backlog of messages that accumulates if the alternate CICS region is held up for any reason. If the data set is too small, you run the risk of the alternate CICS region being unable to read the data set correctly, and thereby becoming incapable of taking over. However, the active CICS region does not write messages to the data set until it has been notified that an alternate CICS region is present (signed on to CAVM), and able to receive them.

The peak in message traffic usually occurs during the “catch-up” phase shortly after the active CICS region detects the presence of the alternate CICS region. You may be able to estimate the amount of space you need from the number of terminal resources you have. The active CICS region sends messages about these resources:

- Installed:
  - VTAM terminals
  - ISC connections
  - MRO connections
  - Consoles
- Autoinstalled terminals
- Bound VTAM terminals that are XRF-eligible

Table 15 lists the sizes of the various messages sent to the data set.

*Table 15. Sizes of messages sent to the XRF message data set*

<b>Type of TCT entry</b>	<b>Bytes per install</b>
The CICS-generated TCT entries (2 only)	629
VTAM terminals	710
Non-3270 devices with pipeline logical units and TASKNO= operand (or TASKLIMIT if RDO) is specified	581 x TASKNO value

Table 15. Sizes of messages sent to the XRF message data set (continued)

Type of TCT entry	Bytes per install
MVS consoles	389
LUTYPE6.2 connection	2083
LUTYPE6.2 mode	169 + (837 x maximum number of sessions)
LUTYPE6.1 connection	226 + (732 x number of sessions)
IRC	237 + (520 x number of sessions)
IRCBCH	240 + (565 x number of sessions)

For VTAM terminals only, you should also make allowance for the following:

Table 16. Additional space requirements (VTAM terminals only)

Bytes per logon	Bytes per logoff	Bytes per signon	Bytes per sign-off
70	35	45	29

The alternate CICS region issues some messages that can help you with your sizing. The following messages issued by the alternate CICS region can give you an idea of the rate of message transfer:

```
DFHTC1041I applid TERMINAL CONTROL TRACKING STARTED
DFHTC1040I applid TERMINAL CONTROL TRACKING RECORDS RECEIVED
DFHTC1043I applid TERMINAL CONTROL TRACKING ENDED - nnn RECORDS RECEIVED
```

The following messages may indicate that the XRF message data set is not large enough:

```
DFHXG6447I NON CRUCIAL XRF MESSAGE(S) DISCARDED
DFHXA6541I XRF HAS FAILED. THE XRF MESSAGE READER IN THE ALTERNATE
SYSTEM HAS FALLEN TOO FAR BEHIND
```

### Crucial and non-crucial messages

The active CICS region classifies its messages as crucial or non-crucial. An example of a crucial message is an autoinstall message that the alternate CICS region must receive if it is to remain eligible to take over. An example of a non-crucial message is a logon message. The alternate CICS region can tolerate the loss of such a message, and the loss only results in some degradation at takeover; no standby session is established for that terminal and it must be logged on again. Install messages that form part of the initial description are also treated as non-crucial, because the active CICS region can try to send them again later, and the alternate CICS region can construct its tables from the CICS catalog if it does not receive a complete initial description.

The active discards non-crucial messages if it decides that sending them may overwrite messages that the alternate CICS region has not yet read, thereby making it ineligible to take over. It issues message DFHXG6447I for the first such discard. The active CICS region always sends crucial messages. If this causes an unread message to be overwritten, the alternate CICS region detects it and terminates after issuing message DFHXA6541I.

### Effect of a full XRF message data set on the active CICS region

The active CICS region is not affected by the state of the XRF message data set. It continues running even when the data set is full; only the alternate CICS region fails. Further, the XRF message data set is only "full" to the alternate CICS region that fails; you can start a new alternate CICS region, using the same XRF message

data set, and the active CICS region resends all the messages for the new alternate CICS region to begin tracking. If the first failure was caused by some unusual condition, you may not need to increase the size of the XRF message data set.

However, if messages DFHXG6447I or DFHXA6541I occur too often, you must stop the active CICS region so that you can change to a larger data set.

## Job control statement for CICS execution

The DDNAME required for CICS execution is DFHXRMSG. The following JCL statement can be used:

```
//DFHXRMSG DD DSN=CICSTS31.CICS.app1id.DFHXRMSG,DISP=SHR
```

---

## Security

To ensure that the integrity and security of your CICS regions and terminal network are not compromised, you must protect your XRF data sets using RACF. When you have done so, give each CICS region CONTROL access to its own pair of data sets. If you are running your XRF systems with an overseer program, make sure that it has READ access to all the CAVM data sets. All other users must be denied access to the data sets.

---

## I/O error handling

While the active CICS region can write its **surveillance signals** successfully to either the XRF control data set or the XRF message data set, it keeps running in spite of I/O errors. However, if the active CICS region has an I/O error while writing a message to the XRF message data set, the alternate CICS region cannot function correctly, so the active CICS region disables it to prevent it from taking over. If the active CICS region is unable to write to either the XRF control data set or the XRF message data set, it can neither disable the alternate CICS region nor keep it properly synchronized, and so the active CICS region fails.

While an alternate CICS region can receive the active CICS region's surveillance signals and tracking messages successfully, in addition to writing its own surveillance signals to either the XRF control data set or the XRF message data set, it keeps running in spite of some types of I/O error. However, an isolated I/O error that would have no effect during tracking, may cause failure of the alternate CICS region if it occurs during takeover.

**Note:** When the active and alternate CICS regions are running in different MVS images, they are not necessarily affected in the same way by the failure of a control unit or channel path that provides access to an CAVM XRF data set.

---

## Chapter 12. Defining user files

This chapter tells you how to define user files and how to access VSAM data sets, BDAM data sets, data tables, and coupling facility data tables.

CICS application programs process files, which, to CICS, are logical views of a physical data set or data table. For data tables, the file provides a view of the data table, which resides either in data space storage or in a coupling facility structure. Except in the case of coupling facility data tables, for which an underlying physical data set is optional, a data table is also associated with a source data set from which the table is loaded. For non-data-table files, the file provides a view of the data set.

A file is identified to CICS by a **file name** of up to eight characters, and there can be many files defined to CICS that refer to the same physical data set or data table. This has the following effect, depending on the type of object the file is defining:

- For non data table files, if more than one file refers to the same data set, each file refers to the same physical data.
- For user-maintained data tables, if more than one file refers to the same data set, each file represents a view of a unique data table.
- For CICS-maintained data tables, if more than one file refers to the same data set, only one can be defined as a CMT. The other files access data from the CMT created by the CMT file definition.
- For coupling facility data tables, if more than one file refers to the same data set, each file represents a view of a unique coupling facility data table in a CFDT pool (unless each file specifies the same tablename and poolname, in which case each they provide a separate view of the same table.

A data set, identified by a data set name (DSNAME) of up to 44 characters, is a collection of data held on disk. CICS file control processes only VSAM or BDAM data. Any data sets referred to by CICS files must be created and cataloged, so that they are known to MVS before any CICS job refers to them. Also, the data sets are usually initialized by being preloaded with at least some data before being used by CICS transactions.

You can use CICS-maintained or user-maintained data tables to improve the performance and function of CICS regions using files that refer to VSAM data sets. Data tables offer a method of constructing, maintaining, and gaining rapid access to data records contained in tables held in data space storage, above 16MB. Each data table is associated with a VSAM KSDS, known as its **source data set**. For more information about data tables, see “Multiple extents and multiple volumes” on page 26.

You can use coupling facility data tables to share data across a sysplex, using the CICS file control API, subject to some restrictions, such as a 16 byte key length.

You can use RLS access mode to share VSAM data sets between CICS application-owning regions throughout a sysplex. See “VSAM record-level sharing (RLS)” on page 114 for further information.

Each of the above methods is discussed under the following topics:

- “VSAM data sets” on page 112
- “BDAM data sets” on page 116
- “Defining data sets to CICS” on page 118

- “Opening VSAM or BDAM files” on page 120
- “Closing VSAM or BDAM files” on page 121
- “XRF considerations” on page 121
- “CICS data tables” on page 123
- “Coupling facility data tables” on page 124.

---

## VSAM data sets

You create a VSAM data set by running the Access Methods Services (AMS) utility program IDCAMS in a batch job, or by using the TSO DEFINE command in a TSO session. The DEFINE command specifies to VSAM and MVS the VSAM attributes and characteristics of your data set. You can also use it to identify the catalog in which your data set is to be defined.

If required, you can load the data set with data, again using IDCAMS. You use the AMS REPRO command to copy data from an existing data set into the newly created one.

You can also load an empty VSAM data set from a CICS transaction. You do this by defining the data set to CICS (by allocating the data set to a CICS file), and then writing data to the data set, regardless of its empty state. See “Loading empty VSAM data sets” on page 113.

When you create a data set, you may define a data set name of up to 44 characters. If you choose not to define a name, VSAM assigns the name for you. This name, known as the data set name (or DSNAME), uniquely identifies the data set to your MVS system.

You can define VSAM data sets accessed by user files under CICS file control as eligible to be backed up while CICS is currently updating these data sets. For more information about backing up VSAM files open for update, see “Backup while open (BWO) of VSAM files” on page 31.

## VSAM bases and paths

You store data in data sets, and retrieve data from data sets, using application programs that reference the data at the record level.

Depending on the type of data set, you can identify a record for retrieval by its key (a unique value in a predefined field in the record), by its relative byte address, or by its relative record number.

Access to records through these methods of primary identification is known as access through the base.

Sometimes you may need to identify and access your records by a secondary or alternate key. With VSAM, you can build one or more alternate indexes over a single base data set, so that you do not need to keep multiple copies of the same information organized in different ways for different applications. Using this method, you create an **alternate index path** (or paths), that links the alternate index (or indexes) with the base. You can then use the alternate key to access the records by specifying the path as the data set to be accessed, that is by allocating the path data set to a CICS file.

When you create a path you give it a name of up to 44 characters, in the same way as a base data set. A CICS application program does not need to know whether it

is accessing your data athrough a path or a base; except that it may be necessary to allow for duplicate keys if the alternate index was specified to have non-unique keys.

## Loading empty VSAM data sets

There are two ways you can load data into an empty VSAM data set. An empty data set can be loaded using either of the following methods:

- Running the AMS utility program, IDCAMS
- Writing records to the data set using CICS transactions

**Note:** Although VSAM imposes some restrictions during initial data set load processing, when the data-set is said to be in **load mode**, these do not affect CICS transactions. For files opened in non-RLS mode, CICS file control “hides” load mode processing from your application programs. For files opened in RLS mode against an empty data set, load mode processing is hidden from CICS by VSAM, and all VSAM requests are allowed.

### Using IDCAMS

If you have a large amount of data to load into a new data set, run the AMS utility program IDCAMS as a batch job, using the REPRO command to copy data from an existing data set to the empty data set. When you have loaded the data set with IDCAMS, it can be used by CICS in the normal way.

**Note:** A data set in VSAM load mode cannot have alternate indexes in the upgrade set. If you want to create and load a data set with alternate indexes, you must use AMS, or some other suitable batch program, to load the data set and invoke BLDINDEX to create the alternate indexes.

### Using CICS applications

If the amount of data to be loaded is small, and there is no upgrade set, you may load an empty data set by using standard CICS file WRITE requests.

When the first write, or series of writes (mass insert), to the file is completed, CICS closes the file and leaves it closed and enabled, so that it will be reopened for normal processing when next referenced. If you attempt to read from a file in load mode, CICS returns a NOTFOUND condition.

## Reuse of data sets

If you define a data set with the AMS REUSE attribute, it may also be emptied of data during a CICS run. This allows it to be used as a work file. When the status of a file referencing the data set is CLOSED and DISABLED (or UNENABLED), you can use the SET EMPTY command, either from an application program using the EXEC CICS command-level interface, or from a master terminal using the master terminal CEMT command. This command sets an indicator in the installed file definition so that when the file is next opened, the VSAM high-used relative byte address (RBA) is set to zero, and the contents of the data set are effectively cleared.

**Note:** If you define a data set to VSAM with the average and maximum record lengths equal, and define a file to CICS with fixed length records to reference that data set, the size of the records written to the data set **must** be of the defined size. For example, if a record in a data set has been read for update, you get errors when rewriting the record if, for example, you:

- Defined the record sizes to VSAM as 250 bytes, with the parameter RECORDSIZE(250 250)
- Defined the file to CICS with the parameter RECFORM=FIXED
- Loaded the data set with records that are only 200 bytes long

## VSAM record-level sharing (RLS)

Record-level sharing (RLS) is an access mode for VSAM data sets supported by DFSMS 1.3 and later releases. RLS enables VSAM data to be shared, with full update capability, between many applications running in many CICS regions.

With RLS, CICS regions that share VSAM data sets can reside in one or more MVS images within an MVS parallel sysplex. This concept, in a parallel sysplex with VSAM RLS supporting a CICSplex, is illustrated in Figure 20.

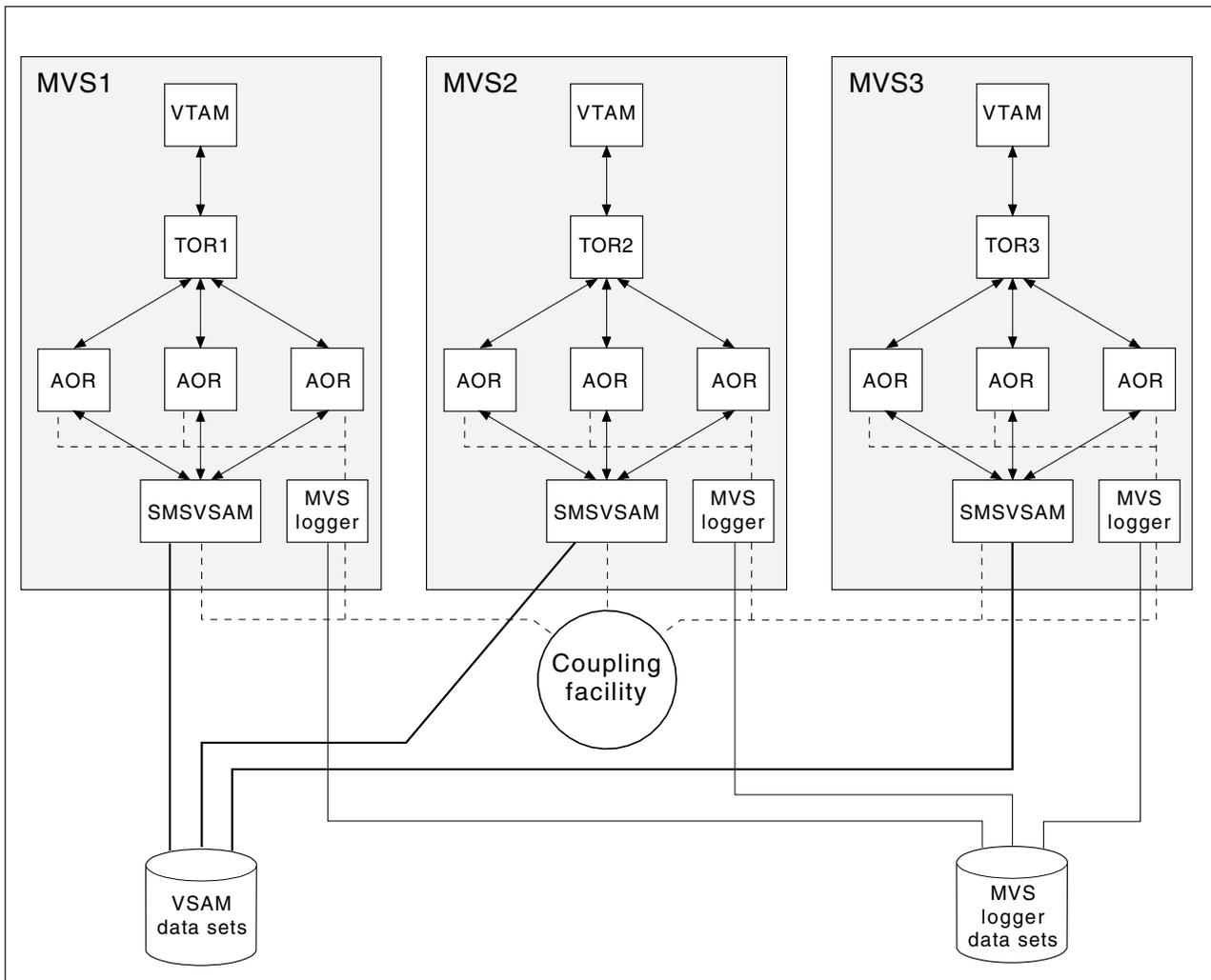


Figure 20. Diagram illustrating a Parallel Sysplex with RLS. This view of RLS shows multiple CICS regions using VSAM RLS, through the services of a SMSVSAM server in each MVS image.

Without RLS support (RLS=NO system initialization parameter), more than one CICS region cannot open the same VSAM data set concurrently using a non-RLS mode (such as LSR or NSR). These access modes mean that to share VSAM data between CICS regions, you must either:

- Use shared data tables,

or

- Allocate the VSAM data sets to one CICS region, a file-owning region (FOR), and function ship file requests from the applications to the FOR using either MRO or APPC connections.

With RLS support, multiple CICS regions can open the same data set concurrently. To use RLS:

- You need a level of DFSMS that supports RLS, and RLS=YES specified as a CICS system initialization parameter
- The CICS regions must all run in the same parallel sysplex
- There must be one SMSVSAM server started in each MVS image
- Specify RLSACCESS=YES in the CICS file resource definition to provide full update capability for data sets accessed by multiple CICS regions.

You can specify RLS access for all files supported by CICS file control, except for the following:

- Key range data sets are not supported.
- VSAM clusters defined with the IMBED attribute are not supported. However, you can remove the IMBED attribute from the cluster definition without loss of function. Use the access method services REPRO function to move the data into a new cluster defined without the IMBED attribute. You can then use RLS access mode for files that reference the new cluster. (IMBED is a performance option that is generally unnecessary with modern caching disk controllers.)
- Opening individual components of a VSAM cluster (which is not supported by CICS for any mode of access).
- Temporary data sets are not supported.
- Key-sequence data sets (KSDS) in relative byte address (RBA) mode (OPTCDE=ADR) are not supported. Application programs that specify the RBA keyword on file control API commands for a KSDS opened RLS mode receive an INVREQ with RESP2 51 exception condition.
- Direct open of alternate index (AIX<sup>®</sup>) data is not supported in RLS access mode. However, path access to data is supported.
- VSAM catalogs and VVDS data sets are not supported.

Although you can specify RLS access for entry-sequenced data sets (ESDS), it is not recommended, because it can have a negative effect on the performance and availability of the data set when you are adding records. (See the *CICS Performance Guide*).

For details of all the steps necessary to set up support for VSAM RLS, see the *CICS Transaction Server for z/OS Installation Guide*.

### **Mixed-mode operation for VSAM data sets**

Generally, you choose which data sets need to be shared and updated in RLS mode by multiple CICS regions. When you have made this choice, you are recommended always to update these data sets in RLS mode.

However, with RLS support, data sets can be shared in mixed access mode, between CICS regions and batch jobs. Mixed access mode means that a data set is open in RLS mode and a non-RLS mode concurrently by different users.

Although data sets can be open in different modes at different times, all the data sets within a VSAM sphere normally should be opened in the same mode. (A

sphere is the collection of all the components—the base, index, any alternate indexes and alternate index paths—associated with a given VSAM base data set.) However, VSAM does permit mixed-mode operations on a sphere by different applications, subject to some CICS restrictions. In the following discussion about mixed-mode operation, references to a data set refer to any component of the sphere.

**SMSVSAM operation of mixed mode:** SMSVSAM permits a data set to be opened in different modes concurrently, by different applications within a sysplex, subject to some sharing rules and limitations:

- A data set that is open in RLS mode to a number of CICS regions can also be opened in non-RLS mode for **read-only** operations
- Read-integrity is not guaranteed for the non-RLS application
- A data set to be opened concurrently in RLS and non-RLS mode must be defined with cross-region SHAREOPTIONS(2)

**CICS restrictions:** You can open a file in RLS mode or non-RLS mode in a CICS region when the referenced data set is already open in a different mode by another user (CICS region or batch job). However, in addition to the above VSAM rules, a data set cannot be open in different modes concurrently within the same CICS region. This ensures that CICS maintains a consistent view of data within the CICS region.

The CICS restrictions operate as follows:

- If a data set is opened in RLS mode in a CICS region, it cannot be opened, through another file, in non-RLS mode in the same CICS region.  
A non-RLS mode file open request fails with message DFHFC0512 if CICS already has the data set open in RLS mode.
- If a data set is opened in non-RLS mode in a CICS region, it cannot be opened, through another file, in RLS mode in the same CICS region.  
An RLS mode file open request fails with message DFHFC0511 if CICS already has the data set open in non-RLS mode.
- If a CICS region has unresolved recovery work for a data set it cannot be opened, through another file, in non-RLS mode in the same CICS region.  
A non-RLS mode file open request fails with message DFHFC0513 if CICS has outstanding recovery work for the data set.

---

## BDAM data sets

CICS supports access to keyed and nonkeyed BDAM data sets. To construct and format such data sets, you use BDAM.

A BDAM data set must contain data before it is used in a CICS run. You load the data set using a batch program that writes the records sequentially. An example of this is Figure 21.

```

//BDAM EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=CICSTS31.bdam.user.file.init,DISP=SHR 1
//SYSUT2 DD DSN=CICSTS31.bdam.user.file,DISP=(,CATLG), 2
// SPACE=(TRK,(1,1)),UNIT=3380,VOL=SER=volid,
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80,DSORG=DA) 3

```

Figure 21. Sample JCL to create and load a BDAM data set

**Notes:**

- 1** The input data set (called SYSUT1 in this example) should be physically sequential and have attributes that are compatible with the DCB for the output data set (called SYSUT2 in this example; see note 3). In particular:
  - If RECFM=F is specified for the output data set, then the input data set must have RECFM=F or RECFM=FB specified, and the value of LRECL should be the same for both data sets.
  - If RECFM=V or RECFM=U is specified for the output data set, then the value of LRECL for the input data set must not be greater than that specified on the output data set.
- 2** When you create a data set, you define a data set name (DSNAME) of up to 44 characters. This data set name uniquely identifies the data set to your MVS system.
- 3** The DCB parameter for the output data set should specify the following:
  - DSORG=DA. This identifies the data set as a BDAM data set.
  - BLKSIZE. This should have the same value as specified for BLKSIZE in the associated file control table (FCT) entry.
  - RECFM. This can take the values F (fixed), V (variable), or U (undefined), and correspond to the first subparameter of the RECFORM operand in the associated FCT entry.

These options are specified on the DFHFCT TYPE=FILE definition. The *CICS Resource Definition Guide* gives information about defining files using DFHFCT TYPE=FILE options. A data set created by this example, and loaded with data such as that shown in Figure 22, would have the following attributes specified in its FCT entry:

- BLKSIZE=80
- LRECL=40
- RECFORM=(FIXED BLOCKED)
- KEYLEN=8

```

RECORD 1 DATA FOR RECORD 1      RECORD 2 DATA FOR RECORD 2
RECORD 3 DATA FOR RECORD 3      RECORD 4 DATA FOR RECORD 4
      ⋮
RECORD98 DATA FOR RECORD 98     RECORD99 DATA FOR RECORD 99
1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

Figure 22. Sample data for loading a BDAM data set

---

## Defining data sets to CICS

Before CICS can open a file referencing a data set, there must be an installed file definition for that file. The definition can be installed either from the CSD or from a file control table (FCT). You can define VSAM files **only** through the CSD, and BDAM files **only** through the FCT. Definitions other than for BDAM in the FCT will not be installed, but are still supported because the DFHFCT macro is used by the migration utility that migrates file definitions to the CSD.

A file is identified by its file name, which you specify when you define the file. CICS uses this name when an application program, or the master terminal operator using the CEMT command, refers to the associated data set.

Each file must also be associated with its data set in one of the following ways:

- Using JCL in the startup job stream
- Using the DSNNAME and DISP parameters of the FILE resource definitions
- Using dynamic allocation with CEMT
- Using dynamic allocation with an application program

### VSAM file limit

The number of VSAM files that can be allocated to a CICS address space is about 10000. However, VSAM maintains a table entry for each file actually opened, and table space limits the number of files opened to about 8189.

## Using JCL

You can define the data set in a DD statement in the JCL of the CICS startup job. The DD name must be the same as the file name that refers to the data set. For example, the following DD statements would correspond to file definitions for the file names VSAM1A and BDAMFILE:

```
//VSAM1A DD DSN=CICSTS31.CICS.vsam.user.file,DISP=OLD
//BDAMFILE DD DSN=CICSTS31.CICS.bdam.user.file,DISP=SHR
```

If you define a data set to CICS in this way it is allocated to CICS by MVS at CICS startup time, and it normally remains allocated until the end of the CICS run. Also, the physical data set is associated with the installed file definition throughout the CICS run.

If you use JCL to define a user data set to the CICS system, the DD statement must not include the FREE=CLOSE operand.

If you use the RLS=CR or RLS=NRI option on your DD statement, it will be ignored. The access mode for the file (RLS or non-RLS) and any read integrity options must be specified in the file definition.

When you are running CICS with XRF, you must specify DISP=SHR for data sets defined in JCL, so that the alternate CICS region can start while the active CICS region's job is also in progress.

### Using the DSNNAME and DISP file resource definition parameters

You can define a data set to CICS by specifying the DSNNAME and DISP operands when you define the file. You can specify these parameters either using RDO for files, or by using DFHFCT macros (BDAM files only). If you want to use DSNNAME

and DISP from the file definition, do **not** provide a DD statement for the data set in the startup job stream, because the attributes in the DD statement will override those in the CICS resource definition.

If you use DSNNAME and DISP on the file definition, CICS allocates the data set dynamically, at the time the first file referencing that data set is opened (that is immediately before the file is opened). At this stage, CICS associates the file name with the data set.

When CICS applications subsequently refer to the data set, they do so by specifying the file name. When you define a data set in this way, it is automatically deallocated by CICS when the file is closed.

For information about using the DSNNAME and DISP parameters, see the SRT system recovery table and the FCT file control table in the *CICS Resource Definition Guide*

### Dynamic allocation using CEMT

You can set the data set name dynamically in an installed file definition by using the master terminal CEMT command:

```
CEMT SET FILE(filename) DSNNAME(datasetname) SHARE|OLD
```

When you use this command, CICS allocates the data set as part of OPEN processing as described above. The data set is automatically deallocated when the last file entry associated with the data set is closed. Before you can dynamically allocate a file using the CEMT command, the file status must be CLOSED, and also be DISABLED or UNENABLED.

This method of defining the data set to CICS allows a file definition to be associated with different data sets at different times. Alternatively, you can close the file and deallocate the data set and then reallocate and open the same file with a different DISP setting. For example, you could do this to enable the physical data set to be shared with a batch program, which reads the data set.

For information about the CEMT SET command, see the *CICS System Programming Reference*.

### Other forms of dynamic allocation

You are recommended to use only those methods of dynamic allocation that are part of CICS file control, and are described in the previous sections.

Do **not** use the CICS dynamic allocation transaction, ADYN, which invokes the sample CICS utility program, DFH99, for dynamic allocation of VSAM and BDAM **user files**. Use of the ADYN transaction may conflict with the dynamic allocation methods used within CICS file control, and can give unpredictable results.

Restrict the use of the ADYN transaction to those data sets not managed by CICS file control, such as auxiliary trace and CICS transaction dump data sets.

For information about the CICS samples, see the *CICS/ESA 4.1 Sample Applications Guide*.

---

## Opening VSAM or BDAM files

Before your application programs can access a file, CICS must first have opened the file using the installed file definition referenced by your program. Part of the process of opening a file is to ensure that the control blocks and buffers required for subsequent processing of the data set are available. If you defined the file to use VSAM local shared resources (LSR), these control blocks and buffers are allocated from the pool of resources. If the LSR pool does not exist at the time of the opening, CICS calculates the requirements and builds the pool before the file is opened. If you defined the file to use nonshared resources, the required control blocks and buffers are allocated by VSAM as part of OPEN processing. If you defined the file to be opened in RLS access mode, VSAM allocates control blocks and buffers in the SMSVSAM address space and associated data set. RLS mode also uses a CF cache structure, to which the data set is bound when the first file that references it is opened.

You may need to access a single VSAM data set either through the base or through one or more paths for different access requests. In this case, CICS uses a separate file definition (that is, a separate file), for each of the access routes. Each file definition must be associated with the corresponding data set name (a path is also assigned a data set name). Each file must be open before CICS can access the file using the attributes in its installed file definition. This is because opening **one** file for a data set that is logically defined as two or more files with different attributes does not mean that the data set is then available for all access routes.

CICS permits more than one file definition to be associated with the same physical data set name. For example, you may want to define files with different processing attributes that refer to the same data set.

CICS allows or denies access to data in a file, depending on whether the state of the file is ENABLED. An **enabled** file that is closed is opened by CICS automatically when the first access request is made. The file remains open until an explicit CLOSE request or until the end of the CICS job.

You can also open a file explicitly by using either of the commands

```
CEMT SET FILE(filename) OPEN
EXEC CICS SET FILE(filename) OPEN
```

When you use one of these commands, the file is opened irrespective of whether its state is enabled or disabled. You may choose this method to avoid the overhead associated with opening the file being borne by the first transaction to access the file.

You can also specify that you want CICS to open a file immediately after initialization by specifying the RDO OPENTIME(STARTUP) attribute (or the FILSTAT=OPENED parameter in the DFHFCT macro). If you specify that you want CICS to open the file after startup, and if the file status is ENABLED or DISABLED, the CICS file utility transaction CSFU opens the file. (CSFU does not open files that are: defined as UNENABLED the status of these remains CLOSED, UNENABLED.) CSFU is initiated automatically, immediately before the completion of CICS initialization. CICS opens each file with a separate OPEN request. If a user transaction starts while CSFU is still running, it can reference and open a file that CSFU has not yet opened; it does not have to wait for CSFU to finish.

---

## Closing VSAM or BDAM files

You can close files with a CLOSE command, with or without the FORCE option.

### Closing files normally

You can close a file explicitly with one of the following commands

```
CEMT SET FILE(filename) CLOSED
EXEC CICS SET FILE(filename) CLOSED
```

The file is closed immediately if there are no transactions using the file at the time of the request. The file is also disabled as part of the close operation, this form of disablement showing as UNENABLED on a CEMT display. This prevents subsequent requests to access the file implicitly reopening it.

A transaction in the process of executing a VSAM or BDAM request, or executing a series of connected requests, is said to be a user of the file. For example, a transaction is a user during the execution of the following requests

```
READ UPDATE ---- REWRITE
STARTBROWSE ---- READNEXT ... ---- ENDBROWSE
```

A transaction is also a user of a file if it completes a recoverable change to the file but has not yet reached a sync point or the end of the transaction.

If there are users at the time of the close request, the file is not closed immediately. CICS waits for all current users to complete their use of the file. The file is placed in an UNENABLING state to deny access to new users but allow existing users to complete their use of the file. When the last user has finished with the file, the file is closed and UNENABLED. If a transaction has made recoverable changes to a file and then suffered a failure during syncpoint, the unit of work is shunted, and the file can be closed at this point.

### Closing files using the FORCE option

You can also close a file using one of the following commands

```
CEMT SET FILE(filename) FORCECLOSE
EXEC CICS SET FILE(filename) CLOSED FORCE
```

Any transactions that are current users of the file are abended and allowed to back out any changes as necessary, and the file is then closed and UNENABLED. A file UNENABLED as a result of a CLOSE request can be reenabled subsequently if an explicit OPEN request is made.

**Note:** Closing a file using the FORCE option causes tasks of any current users of the file to be terminated immediately by the CICS task FORCEPURGE mechanism. Data integrity is not guaranteed with this mechanism. In some extreme cases (for example, if an error occurs during backout processing), CICS might terminate abnormally. For this reason, closing files using the FORCE option should be restricted to exceptional circumstances.

---

## XRF considerations

For both VSAM: and BDAM files

- The active and alternate CICS region must refer to the same data sets. To ensure that they do, you can define the files using the DSNAME and DISP parameters, and allow CICS to allocate the data sets dynamically. Omitting DD

statements in the job streams for the active and alternate CICS regions minimizes the risk of inconsistency in data set naming.

- The alternate CICS region does not open the data sets before takeover occurs.
- If the data sets are allocated at job step initiation, the JCL defining the data sets must specify DISP=SHR.

---

## CICS data tables

A data table is defined by means of the CEDD DEFINE FILE command. When a table is opened, CICS builds it by extracting data from the tables corresponding source VSAM data set and loading it into an MVS data space owned by the CICS data tables server region, and constructing an index in CICS virtual storage above the 16MB line. The commands used to access these tables are the file control commands of the CICS application programming interface (API).

For information about defining CICS data tables, see the *CICS Resource Definition Guide*. For programming information about the file control commands of the application programming interface, see the *CICS Application Programming Reference*. CICS supports two types of data table :

- **CICS-maintained data tables** that CICS keeps in synchronization with their source data sets.
- **User-maintained data tables** that are completely detached from their source data sets after being loaded.

For either type, a global user exit can be used to select which records from the source data set should be included in the data table.

For programming interface information about global user exits, see the *CICS Customization Guide*. For further information on CICS data tables, see the *CICS Shared Data Tables Guide*.

## Opening data tables

A data table must be opened before its entries can be accessed by an application. You can open a data table explicitly with an OPEN request, implicitly on first reference, or by the CSFU task just after startup, if OPENTIME(STARTUP) was specified in the file definition. When a data table is opened, CICS reads the complete source data set, copying the records into a data space and building an index.

A global user exit can be invoked for each record copied into the data table. This copying is subject to any selection criteria of the user-written exit.

The commands used to open data tables, and the rules and options concerning their implicit and immediate opening are the same as those described in “Opening VSAM or BDAM files” on page 120.

## Loading data tables

A data table is built automatically when it is opened. An index is constructed to provide rapid access to the records. See the *CICS Shared Data Tables Guide* for more details.

For a user-maintained data table, the ACB for the source data set is closed when loading has been completed. The data set is deallocated if it was originally dynamically allocated and there are no other ACBs open for it.

## Closing data tables

You can close a data table with a CLOSE command, with or without the FORCE option. When a data table is closed, the data space storage that was used to hold the records and the address space storage used for the associated index, is freed as part of the CLOSE operation.

The commands used to close data tables, and the rules concerning current users of a data table are the same as those described in “Closing VSAM or BDAM files” on page 121.

## XRF considerations

After an XRF takeover, a data table must be reloaded from its source data set when the data table is opened. For a CICS-maintained data table, the effect is to restore the data table to its final state in the previous active CICS region, because CICS keeps data tables and source data sets in step. For a user-maintained data table, the relationship of the current contents of the source data set to the contents of the data table when the previous active CICS region terminated is application-dependent.

---

## Coupling facility data tables

Coupling facility data tables provide a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. CICS coupling facility data table support is designed to provide rapid sharing of working data within a sysplex, with update integrity. The data is held in a coupling facility, in a table that is similar in many ways to a shared user-maintained data table. This section describes how to define the resources required for coupling facility data tables in an MVS coupling facility resource management (CFRM) policy.

Because read access and write access have similar performance, this form of table is particularly useful for scratchpad data. Typical uses might include sharing scratchpad data between CICS regions across a sysplex, or sharing of files for which changes do not have to be permanently saved. There are many different requirements for scratchpad data, and most of these can be implemented using coupling facility data tables. Coupling facility data tables are particularly useful for grouping data into different tables, where the items can be identified and retrieved by their keys. For example, you could use a field in a coupling facility data table to maintain the next free order number for use by an order processing application, or you could maintain a list of the numbers of lost credit cards in a coupling facility data table.

## Comparison with user-maintained data tables

To an application, a coupling facility data table (CFDT) appears much like a sysplex-wide user-maintained data table, because it is accessed in the same way using the file control API. However, in a CFDT there is a maximum key-length restriction of 16 bytes.

## Coupling facility data table models

There are two models of coupling facility data table:

- The contention model, which gives optimal performance but generally requires programs written to exploit it. This is because the CHANGED condition code (indicating that the data has been changed since the application program issued a read-for-update request) is specifically for this model, and programs not written for the contention model may not be able to handle this condition correctly. The CHANGED response can occur on a REWRITE or DELETE command. There is also a situation with the contention model in which the NOTFND response can be returned on a REWRITE or DELETE.

This model is non-recoverable: CFDT updates are not backed out if a unit of work fails.

- The locking model is API-compatible with programs that conform to the UMT subset of the file control API (this subset is nearly, but not quite, the full file control API).

This model can either be:

- **Non-recoverable:** locks do not last until syncpoint, and CFDT updates are not backed out if a unit of work fails, or
- **Recoverable:** coupling facility data tables are recoverable in the event of a unit of work failure and in the event of a CICS region failure (in that updates made by units of work that were in-flight at the time of the CICS failure are backed out).

The recoverable locking model supports in-doubt and backout failures: if a unit of work fails when backing out an update to the CFDT or if it fails in-doubt during syncpoint processing the locks are converted to retained locks and the unit of work is shunted.

You specify the model you want for each table on its file resource definition, enabling different tables to use different models.

## Coupling facility data table structures and servers

Coupling facility data tables are held in coupling facility structures. Access to a coupling facility data table is through a named server, which can be thought of as similar to a shared data tables file-owning region (see the *CICS Shared Data Tables Guide* for information about shared data tables support). Coupling facility data tables support allows you to separate related groups of coupling facility data tables by storing them in separate pools. For example, you might want to have one pool for production and another for test.

A coupling facility data table pool is a coupling facility list structure, and access to it is provided by a coupling facility data table server. Within each MVS image, there must be one CFDT server for each CFDT pool accessed by CICS regions in the MVS image. The names of the servers are formed by adding the prefix DFHCF to the pool name, giving *DFHCF.poolname*. Coupling facility data table pools are defined in the coupling facility resource management (CFRM) policy. The pool name is then specified in the start-up JCL for the table server.

### Access using file control API

A coupling facility data table is accessed from CICS through file control commands. The file name specified on the command indicates the name of the table and pool in which it resides. The table name is either specified on the file definition or is the same as the file name, and the pool name is specified on the file definition. The table is uniquely identified by the pool name and table name, so that two tables with the same name may exist in different pools, and will be entirely separate entities.

### Automatic connection to coupling facility data table pools

CICS automatically connects to the coupling facility data table server for a given pool the first time that a coupling facility data table within that pool is referenced. CICS also automatically reconnects to the coupling facility data table server when the server restarts after a failure.

Coupling facility data table servers are protected against misuse by CICS regions that call them, thus ensuring system integrity. In particular, protection is provided to prevent calling region from being able to modify sensitive parameters to authorized functions.

Likewise, CICS is protected from any side effects if a coupling facility data table server fails. If a CICS region issues a file control request to a coupling facility data table server that has failed, the resulting MVS abend is trapped and returned to the application program as a SYSIDERR condition.

### **Creating coupling facility data tables**

CICS automatically creates a coupling facility data table (CFDT) when a first reference requires the CFDT to be opened. This CFDT is then used by the same region, or other CICS regions, that issue subsequent open requests of other files that name the same coupling facility data table.

CICS can optionally load the coupling facility data table automatically from a source VSAM (KSDS) data set when it is first opened. Unlike user-maintained data tables, with coupling facility data tables you can specify that there is no associated source data set, allowing you to create an empty CFDT.

Your application programs have access to a coupling facility data table as soon as it is created, although there are some restrictions on the keys that can be accessed while data is being loaded.

When a CFDT is loaded, it becomes an independent entity, separate from the behavior of the CICS regions that access the table or caused the table to be loaded. Even when all CICS regions have terminated, either normally or abnormally, a CFDT continues to remain in the coupling facility until you either take explicit action to delete the structure, or the coupling facility. You can delete the CFDT contents or structure with a `MODIFY cfdt_server, DELETE TABLE=name` command.

### **Administering coupling facility data tables**

CICS provides some utility functions that allow you to obtain, from a coupling facility data table server, summary information and periodic statistics on coupling facility data tables defined within a pool. This information is designed to help you to administer coupling facility data table pools, and to help you to evaluate capacity. See “Coupling facility data table server parameters” on page 386 for details.

## **Defining a coupling facility data table pool**

You define the list structure for a coupling facility data table in a coupling facility resource manager (CFRM) policy in a sysplex couple data set. A coupling facility data table pool, containing one or more coupling facility data tables, is accessed by CICS through a server region using cross-memory services. (See Chapter 25, “Setting up and running a coupling facility data table server,” on page 383 for information about setting up and starting a coupling facility data table region.)

From the application point of view, a pool and its server are similar to a file-owning region, and the pool can contain any number of tables provided that each one has a unique table name.

Before a coupling facility data table server can use its pool, the active CFRM policy must contain a definition of the list structure to be used for the pool. To achieve this, add a statement that specifies the list structure to a selected CFRM policy, and then activate the policy.

The CFRM structure definition specifies the size of the list structure and the preference list of coupling facilities in which it can be stored. You create the name of the list structure for a coupling facility data table pool by adding the prefix `DFHCFLS_` to the pool name, giving `DFHCFLS_poolname`.

## Using IXCMIAPU to update a policy

To update an administrative policy in the CFRM couple data set, use the administrative data utility, IXCMIAPU. The utility adds or changes policy data in the administrative policies only: it does not change any information in the system's copy of the active CFRM policy. For an example of a job to run this utility, see member IXCCFRMP in the SYS1.SAMPLIB library. An example of a policy statement for a coupling facility data table pool is shown in Figure 23.

```
STRUCTURE NAME(DFHCFLS_PRODCT1)
SIZE(1000)
INITSIZE(500)
PREFLIST(FACIL01,FACIL02)
```

Figure 23. Example of defining a coupling facility data table structure

**Activating a CFRM policy:** When you have defined the list structure in a CFRM policy, activate the policy using the MVS command SETXCF START,POLICY,POLNAME=*polycyname*,TYPE=CFRM. Note that activating a CFRM policy that contains a definition of a list structure does not create the structure. It is created the first time an attempt is made to connect to it, which occurs when the first coupling facility data table server that refers to the corresponding pool is started.

When the server creates a list structure, it is allocated with an initial size, which can be increased up to a maximum size as specified in the CFRM policy. All structure sizes are rounded up to the next multiple of 256KB at allocation time. Provided that space is available in the coupling facility, you can dynamically expand a list structure from its initial size up to its maximum size, or contract it to free up coupling facility space for other purposes. Note that if the initial structure allocation becomes full, the structure does not expand automatically, even if the structure allocated is less than the specified maximum size. To expand a list structure when the allocation becomes full, you can expand it (up to its maximum size) using the following SETXCF command:

```
SETXCF START,ALTER,STRNAME=DFHCFLS_poolname,SIZE=nnnn
```

Note that if you dynamically increase the size of a list structure in this way, also update the INITSIZE parameter in the policy to reflect the new size, so that the structure does not revert to its original size if you subsequently recreate or reload it.

## Calculating the structure size for a pool

A coupling facility structure contains not only stored data but also the information needed to manage and access that data, in a similar way to a key-sequenced data set. The data for each entry in the coupling facility is stored as a chain of fixed size (usually 256-byte) elements, which means that the exact length for variable-length data has to be stored separately. CICS does this by including a length prefix in the stored data, so space calculations have to allow for each entry using a whole number of elements. The amount of internal control information depends on the level of functionality and performance of the coupling facility control code for the current CFLEVEL. The storage requirements can increase for a higher CFLEVEL. For more information about how the coupling facility structure storage is organized, see “Coupling facility storage management” on page 424.

The easiest way to calculate accurate structure storage requirements is to use the web-based IBM CFSizer tool at <http://www-1.ibm.com/servers/eserver/zseries/cfsizer/>. The tool takes these factors into account and communicates with a

# coupling facility at a current CFLEVEL to perform the calculation. You need to enter  
# some minimum input to get an accurate storage calculation as follows:

# **MAXQUEUES**

# specifies the maximum number of data lists that are reserved when the  
# structure list is allocated by CICS. This determines the maximum number of  
# large queues that can be stored in the structure and corresponds to the  
# MAXQUEUES server parameter. A large queue is where the total size of  
# the data items exceeds 32K. It is stored in a separate list in the structure.  
# Although you should specify a large enough number, specifying an  
# excessively large number will use up an unnecessary amount of coupling  
# facility storage for unused preallocated list headers. The valid range is from  
# 1 to 999999. The default is 1000.

# **Average rounded item size**

# specifies the average amount of storage required for each TS queue item.  
# Note that each item has a two-byte length prefix and is stored as one or  
# more 256-byte elements. This number should range from 1 to 32768. This  
# value determines the entry to element ratio that is used to calculate the  
# required structure size.

# If all queue items are approximately the same size, calculate this value by  
# taking the average data size, adding two, and rounding up to the next  
# multiple of 256. If queue items are different sizes, round up each size first  
# before taking the average. For example, if half of the items are 100 bytes  
# and half are 300 bytes, then the rounded sizes are 256 and 512. The  
# average rounded item size is half way between those values, 384, which is  
# more accurate than taking the overall average item size of 200 and  
# rounding it up to 256.

# **Total number of items in all queues**

# specifies the total number of entries in all of the TS queues.

# **Target Usage Percent**

# specifies the percentage of the structure space which the given total  
# number of items are expected to use. Specify a number in the range of 1 to  
# 100. The default is 75. This leaves some free space for temporary  
# expansion, and to give time to expand the structure in response to warning  
# messages (which normally start at 80%) if the initial free space is not  
# enough.

# **Max Expansion Percent**

# specifies the percentage that the structure can expand. If a non-zero value  
# is specified here, the maximum structure size will be greater than the initial  
# structure size by an amount sufficient to allow for a further expansion of the  
# total amount of data by this percentage. For example, if the value 200 is  
# specified, the initial size will be enough to store the specified total number  
# of items, but the maximum size will be enough to store three times that  
# number of items.

# Alternatively you can use the following calculation, which applies up to CFLEVEL 11  
# of the coupling facility. If you have a coupling facility at CFLEVEL 12, you need to  
# change the calculation to add 10% and 2MB to the sizings in the formula below.

### Storage calculations

Data entry size =  $(170 + (\textit{average record data size}^1))$   
+ 5% extra for control information

<sup>1</sup> Average record data size must have a 2-byte prefix added  
and be rounded up to a multiple of 256 bytes.

Total size = 200KB  
+  $(\textit{number of tables} \times 1\text{KB})$   
+  $(\textit{number of records in all tables} \times \textit{data entry size})$

The above calculation assumes that the structure is allocated at its maximum size. If it is allocated at less than its maximum size, the same amount of control information is still required, so the percentage of space occupied by control information is correspondingly increased. For example, if a structure is allocated at one third of its maximum size, the overhead for control information increases to around fifteen per cent.

**Note:** The list headers for a table use about two-thirds of the 1KB in the storage formula shown above. If you let the MAXTABLES parameter take the default value, about 700KB of the list structure size is taken up by the list headers alone.

The calculation determines the size of a structure that would be approximately 100% full for the specified level of usage. For practical operation, however, a reasonable proportion of free space must be available, not only to minimize the risk of the structure becoming totally full but also to avoid triggering low space warning messages and additional activity to alter entry to element ratios. The maximum normal usage should therefore aim to be about 75% of the structure size. Expected usage values should therefore be adjusted upwards to allow for the required amount of free space (for example, by about one third to aim for 75% usage).

For information about the reserved space parameters you can use to enable the server to avoid a structure full condition, see “Reserved space parameters” on page 393 and “Avoiding structure full conditions” on page 394.



## Chapter 13. Defining the CDBM GROUP command data set

This chapter describes the CICS DFHDBFK file. The DFHDBFK file, used by the CDBM transaction to provide a repository for stored groups of DBCTL commands, is a VSAM key-sequenced data set (KSDS).

You can create the DFHDBFK data set by running an IDCAMS job, an example of which is shown in Figure 24. You can use this job to load some IMS™ commands, or you can use the maintenance function within the CDBM transaction.

```
//DBFKJOB JOB 'accounting information',name,MSGCLASS=A
/*
//DBFKDEF EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYSIN DD *
DELETE CICSTS31.CICS.DFHDBFK
SET MAXCC=0
DEFINE CLUSTER (
    NAME( CICSTS31.CICS.DFHDBFK ) -
    INDEXED -
    RECORDS(100 20) -
    KEYS(22,0) -
    RECORDSIZE(1428 1428) -
)
INDEX (
    NAME( CICSTS31.CICS.DFHDBFK.INDEX ) -
    CONTROLINTERVALSIZE(512) -
)
DATA (
    NAME( CICSTS31.CICS.DFHDBFK.DATA ) -
    CONTROLINTERVALSIZE(2048) -
)
/*
/* The next two job steps are optional.
/*
//DBFKINID EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE CICSTS31.CICS.DBFKINIT
/*
//DBFKINIF EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=CICSTS31.CICS.DBFKINIT,DISP=(NEW,CATLG),
// UNIT=dbfkunit,VOL=SER=dbfkvol,SPACE=(TRK,(1,1)),
// DCB=(RECFM=FB,LRECL=40,BLKSIZE=6160)
```

Figure 24. Sample job to define and initialize the DFHDBFK data set (Part 1 of 2)

```

/* Place the definitions you want to load after SYSUT1. For example:
//SYSUT1 DD *
SAMPLE    DIS      DB DI21PART
SAMPLE    STA      DB DI21PART
SAMPLE    STO      DB DI21PART
/*
//SYSIN   DD *
GENERATE MAXFLDS=1
RECORD FIELD=(40)
/*
//DBFKLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYS01    DD DSN=CICSTS31.CICS.DBFKINIT,DISP=SHR
//DFHDBFK DD DSN=CICSTS31.CICS.DFHDBKF,DISP=SHR
//SYSIN   DD *
        REPRO INFILE (SYS01)           -
              OUTFILE (DFHDBFK)
/*
//

```

where *dbfkvol* is the volume on which the DFHDBFK data set is to be created, and *dbfkunit* is the unit type for that volume.

Figure 24. Sample job to define and initialize the DFHDBFK data set (Part 2 of 2)

## Job control statements for CICS execution

If you define the DFHDBFK data set using the sample JCL shown in Figure 24 on page 131, the data definition statement for the CICS execution is as follows:

```
//DFHDBFK DD DSN=CICSTS31.CICS.DFHDBFK,DISP=SHR
```

Alternatively, if you want to use dynamic file allocation, add the fully-qualified data set name to the DFHDBFK file resource definition.

## Record layout in the CDBM GROUP command file

Each record in the DFHDBFK file may be up to 1428 characters long, as follows:

field	length	content	description
1	12	Group	a 12-character field containing your chosen name for this group. The acceptable characters are A-Z 0-9 \$ @ and #. Leading or embedded blanks are not allowed, but trailing blanks are acceptable.
2	10	IMS Command	a 10-character field containing any of the IMS command verbs that are valid for CDBM (see the <i>CICS IMS Database Control Guide</i> for details). Leading or embedded blanks are not allowed, but trailing blanks are acceptable. <b>Note:</b> The validity of the IMS command verb is not checked by CDBM. Invalid values will be reported by IMS when the command is attempted.

field	length	content	description
3	1406	IMS Command parameters	Up to 1406 characters of parameters appropriate to the chosen IMS command verb. (This will often consist of lists of databases.) <b>Note:</b> Wildcard characters may not be used in the parameters stored in the CDBM Group command file. This is unlike the other functions of the CDBM transaction which permit the use of wildcard characters to describe multiple similarly named databases.



---

## Chapter 14. Defining the CMAC messages data set

This chapter describes the VSAM key-sequenced data set (KSDS) called DFHMACD. DFHMACD is used by the CMAC transaction to provide online descriptions of the CICS messages and codes.

You can create the DFHMACD data set and load it with the CICS-supplied messages and codes data by running the DFHMACI job. Some IBM-supplied service may include changes to CICS messages and codes, and associated changes to the DFHMACD data set. You can apply such service changes to the DFHMACD data set by running the DFHMACU job.

For more information about the DFHMACI and DFHMACU jobs, see the *CICS Transaction Server for z/OS Installation Guide*.

### Notes:

1. The DFHMACD data set is accessed by the file CMAC, managed by CICS File Control. You must create a definition for this file in the CSD or FCT. The CICS-supplied definition for the CMAC file and other resources needed by the CICS messages facility are in the CSD group DFHMAC. The CICS IVPs have a DD statement for the CMAC file, but for dynamic allocation you should copy the supplied resource definition for the CMAC file and add the DSNAMES option.
2. To use the CICS messages facility in your CICS region, you must create your own CSD group list to include the CICS-supplied group list DFHLIST, the DFHMAC group for the CICS messages facility, and any other groups of resources that your CICS region needs. You must specify this group list by using the system initialization parameter GRPLIST when you start up your CICS region.
3. You should specify the DFHMAC group of resources for the CICS messages facility only in those CICS regions that need to use the facility; for example on some terminal-owning regions, but perhaps not on data-owning regions.

---

## Job control statements to define and load the messages data set

Before its first use, the DFHMACD data set should be defined and loaded as a VSAM key sequenced data set (KSDS). The sample job in Figure 25 on page 136 shows you how to do this.

**Note:** You can define and load the DFHMACD data set by running the DFHMACI job.

```

//CMACJOB JOB 'accounting information',name,MSGCLASS=A
//CMACDEF EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYSIN DD *
DELETE CICSTS31.CICS.DFHCMACD
SET MAXCC=0
DEFINE CLUSTER (
        NAME( CICSTS31.CICS.DFHCMACD ) -
        CYL(2,1) -
        KEYS( 9 0 ) -
        INDEXED -
        VOLUME ( cmacvol ) -
        RECORDSIZE( 8192 30646 ) -
        FREESPACE( 5 5 ) -
        SHAREOPTIONS( 2 ) -
        )
INDEX (
        NAME( CICSTS31.CICS.DFHCMACD.INDEX ) -
        )
DATA (
        NAME( CICSTS31.CICS.DFHCMACD.DATA ) -
        )
/*
//CMACLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYS01 DD DSN=CICSTS31.CICS.SDFHMSG(SDFHMACD),DISP=SHR
//DFHMACD DD DSN=CICSTS31.CICS.DFHCMACD,DISP=SHR
//SYSIN DD *
        REPRO INFILE (SYS01) -
                OUTFILE (DFHMACD)
/*
//

```

where cmacvol is the volume on which the DFHMACD data set is to be created.

Figure 25. Sample job to define and initialize the CMAC data set

## Job control statements for CICS execution

If you defined the messages data set using the sample job shown in Figure 25, the data definition statement for the CICS execution is:

```
//DFHMACD DD DSN=CICSTS31.CICS.DFHCMACD,DISP=SHR
```

---

## Chapter 15. Defining the EJB data sets

This chapter describes the VSAM key-sequenced data sets you need for CICS EJB support. These data sets are:

#  
#

- The EJB directory, DFHEJDIR
- The EJB object store, DFHEJOS

These data sets, and the IDCAMS definition statements for creating them, are described in the following topic.

---

### Defining EJB directory and object store data sets

DFHEJDIR is a file containing a request streams directory that is shared by all the regions (listeners and AORs) in a logical EJB server.

DFHEJOS is a file of passivated stateful session beans. It is shared by all the AORs in the logical EJB server.

The EJB directory and object store data sets are CICS file-control-managed data sets that require resource definitions in the CSD. Because the data sets have to be shared by all the CICS regions that form a logical EJB server, you can define each of them to CICS file control as:

- An ordinary VSAM data set, to be opened in either LSR or NSR mode (that is, with the LSRPOOLID attribute specified with a pool number, or as NONE). In this case, the data set has to be owned by one CICS region (a file-owning region) to which the others in the EJB server can function ship file requests.

Definitions for both data sets are supplied in the CICS CSD group called DFHEJVS, with the LSRPOOLID default value of 1. Note that this CSD group is supplied *unlocked*, so that you can edit the file definitions to add details such as the data set name for dynamic allocation, a specific LSRPOOLID to match an explicit LSRPOOL resource definition, or the remote system attributes.

- A VSAM data set to be opened in RLS mode. Definitions for both data sets are supplied in the CICS CSD group called DFHEJVR with RLSACCESS(YES) specified. Note that this CSD group is supplied *unlocked*, so that you can edit the file definitions to add attributes such as the data set name for dynamic allocation.
- A coupling facility data table. Definitions for both data sets are supplied in the CICS CSD group called DFHEJCF. Note that this CSD group is supplied *unlocked*, so that you can edit the file definitions to modify the CFDT details, such as the pool name and the table name.

Figure 26 on page 138 shows an example of the JCL you can use to define an EJB directory data set.

```

//EJBDIR  JOB accounting info,,CLASS=A
//DEFGCD  EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN   DD   *
/*
/* DEFINE A DIRECTORY          */
/* DATASET                     */
/*                             */
DEFINE CLUSTER(NAME(CICSTS31.CICS.DFHEJDIR)-
  INDEXED-
  LOG(UNDO)- 1
  CYL(2 1)-
  VOLUME(valid)- 2
  RECORDSIZE( 1017 1017 )- 3
  KEYS( 16 0 )-
  FREESPACE ( 10 10 )-
  SHAREOPTIONS( 2 3 ))-
DATA (NAME(CICSTS31.CICS.DFHEJDIR.DATA) -
  CONTROLINTERVALSIZE(1024)) -
INDEX (NAME(CICSTS31.CICS.DFHEJDIR.INDEX))
/*
//*

```

Figure 26. Example JCL to define an EJB directory data set

**1** The backout recovery attribute is defined in the ICF catalog so that the data set defined by this job can be used in either RLS or non-RLS mode. For data sets used in RLS mode, the recovery attributes must be defined in the ICF catalog, and they override any that are specified in the CICS file resource definition. The EJB directory data set must be defined as recoverable.

**2** Specify your own value for the VOLUME parameter, or remove it altogether if you are using SMS-managed storage.

**3** The average and maximum record size must both be defined as 1017 bytes.

**Note:** You should not need to change any of the definition values shown in Figure 26: DFHEJDIR only contains one record, its control record. However, ensure that run-time settings are specified (such as BUFND, BUFNI and STRNO subparameters on the AMP parameter on the DD statement, or the equivalent in the CICS file resource definition) to support the maximum number of requests that could be active at any one time.

Figure 27 on page 139 shows an example of the JCL you can use to define an EJB object store.

```

//EJBOS    JOB accounting info,,CLASS=A
//DEFGCD   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD   *
/*          */
/* DEFINE AN OBJECT STORE          */
/* DATASET                          */
/*          */
DEFINE CLUSTER(NAME(CICSTS31.CICS.DFHEJOS) -
              INDEXED -
              LOG(NONE) -
              RECORDS(1000,100) -
              VOLUME(valid) -
              RECORDSIZE( 8185 8185 ) -
              KEYS( 16 0 ) -
              FREESPACE ( 10 10 ) -
              SHAREOPTIONS( 2 3 )) -
DATA (NAME(CICSTS31.CICS.DFHEJOS.DATA) -
      CONTROLINTERVALSIZE(8192)) -
INDEX (NAME(CICSTS31.CICS.DFHEJOS.INDEX))
/*
//*

```

Figure 27. Example JCL to define an EJB object store data set

**1** The backout recovery attribute is defined in the ICF catalog so that the data set defined by this job can be used in either RLS or non-RLS mode. For data sets used in RLS mode, the recovery attributes must be defined in the ICF catalog, and they override any that are specified in the CICS file resource definition. The EJB object store data set must be defined as non-recoverable.

**2** The number of records that the file can hold. VSAM automatically calculates space requirements.

**3** Specify your own value for the VOLUME parameter, or remove it altogether if you are using SMS-managed storage.

**4** The average and maximum record size are both shown as 8185 bytes in this sample job, but you will need to calculate your own size as described in “Determining the object store space requirements.”

## Determining the object store space requirements

Follow these guidelines when calculating the size of your object store data set:

1. Find the average size of a stateful bean. To do this you can write a method that serializes a bean and writes it to a file. The filesize is the size of the bean. Repeat for each bean, and then calculate the average filesize.

**Note:** The size of the bean depends on what state the bean has. For example, if a stateful bean contains a hashtable populated with 100 KB of non-transient data then the serialized object must also contain this information.

2. Round the figure at Step 1 up to nearest multiple of 512 bytes (subject to a minimum of 1 024 bytes).
3. Estimate the maximum number of records, and use this as the primary allocation in the RECORDS parameter ( see **2** ).
4. Estimate the requirements for the secondary allocation, for example by halving the primary allocation.

Modify the supplied IDCAMS step in DFHDEFDS to create an object store that meets your needs.

## # Chapter 16. Defining the WS-AT data set

# The data set you require to enable CICS Web Services Atomic Transaction (WS-AT) support is the WS-AT directory data set, DFHPIDIR. The WS-AT directory data set, DFHPIDIR, is a file containing a mapping between contexts and tasks.

# The WS-AT directory and object store data set, DFHPIDIR, is a CICS file-control-managed data set that requires a resource definition in the CSD. Because the data set is shared across CICS regions that together provide a WS-AT capable Web service provider, you can define it to CICS file control in these ways:

# • An ordinary VSAM data set, to be opened in either LSR or NSR mode; that is, with the LSRPOOLID attribute specified with a pool number, or as NONE. In this case, the data set has to be owned by one CICS region (a file-owning region) to which the other regions can function ship file requests.

# A definition for the data set is supplied in the CICS CSD group DFHPIVS, with the LSRPOOLID default value of 1. Make a copy of this CSD group, rename it, and edit the file definitions to add details such as the data set name for dynamic allocation, a specific LSRPOOLID to match an explicit LSRPOOL resource definition, or the remote system attributes.

# • A VSAM data set to be opened in RLS mode. A definition for the data set is supplied in the CICS CSD group DFHPIVR with RLSACCESS(YES) specified. Make a copy of this CSD group, rename it, and edit the file definitions to add attributes such as the data set name for dynamic allocation.

# • A coupling facility data table. A definition for the data set is supplied in the CICS CSD group DFHPICF. Make a copy of this CSD group, rename it, and edit the file definitions to modify the CFDT details, such as the pool name and the table name.

# Figure 28 shows the statements used to define DFHPIDIR.

#

```
# DEFINE CLUSTER(NAME(@dsindex@.CICS@regname@.DFHPIDIR)-  
# INDEXED-  
# LOG(UNDO)- 1  
# CYL(2 1)-  
# VOLUME(@dsvol@)- 2  
# RECORDSIZE( 1017 1017 )- 3  
# KEYS( 16 0 )-  
# FREESPACE ( 10 10 )-  
# SHAREOPTIONS(2 3 )-  
# DATA (NAME(@dsindex@.CICS@regname@.DFHPIDIR.DATA)-  
# CONTROLINTERVALSIZE(1024)) -  
# INDEX (NAME(@dsindex@.CICS@regname@.DFHPIDIR.INDEX))  
#  
#  
#
```

Figure 28. Example JCL to define the WS-AT directory data set

# **1** Define the backout recovery attribute in the ICF catalog, so that the data set defined by this job can be used in either RLS or non-RLS mode. For data sets used in RLS mode, you define the recovery attributes in the ICF catalog, and these attributes override any that are specified in the CICS file resource definition. You must define the PI directory data set as recoverable.

# **2** Specify your own value for the VOLUME parameter, or remove it completely if you are using SMS-managed storage.

# **3** The default record size is 1 KB, which can be changed.

# **Note:** You do not change any of the definition values shown in Figure 28 on page  
# 141. DFHPIDIR contains only one record, its control record. However, ensure  
# that runtime settings are specified (such as BUFND, BUFNI; and STRNO  
# subparameters on the AMP parameter on the DD statement, or the  
# equivalent in the CICS file resource definition) to support the maximum  
# number of requests that might be active at any one time.

---

## Chapter 17. Setting up the debugging profiles data sets

Application programmers who use certain debugging tools with CICS create debugging profiles which are stored in a VSAM key sequenced data set with an alternative index.

To set up the debugging profiles data sets:

1. Use the IDCAMS utility to create and initialize the VSAM data sets.
2. Create file definitions for the data sets. The data sets can be shared by more than one CICS region, and can be defined as:

### **VSAM RLS files**

Define VSAM RLS files when you want to share profiles between several CICS regions in the same sysplex

### **VSAM non-RLS files**

Define VSAM non-RLS files when you do not need to share profiles between regions in the same sysplex

### **Remote files**

Define remote files when you want to use profiles that are stored in a region that is connected using MRO or ISC.

---

## Creating the debugging profiles data sets

Use the IDCAMS utility to create and initialize the following VSAM data sets:

### **DFHDPFMB**

The debugging profiles base data set.

### **DFHDPFMP**

The debugging profiles path data set.

### **DFHDPFMX**

The debugging profiles alternate index data set.

Use the JCL in Figure 29 on page 144.

```

//DPFM JOB 'accounting information',name,MSGCLASS=A
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    DELETE CICSTS31.CICS.DFHDPFMB

    DEFINE CLUSTER (RECORDS(1000)-
        NAME (CICSTS31.CICS.DFHDPFMB) -
        SHAREOPTIONS(2 3) -
        LOG(NONE) -
        VOLUME (&DSVOL) -
        IXD) -
    DATA -
        (RECSZ(2560,2560) -
        CONTROLINTERVALSIZE(3072) -
        NAME (CICSTS31.CICS.DFHDPFMB.DATA) -
        KEYS(17 1) -
        FREESPACE(10 10) -
        BUFFERSPACE (8192)) -
    INDEX -
        (NAME(CICSTS31.CICS.DFHDPFMB.INDX))
//INITDP EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    REPRO INFILE ( SYS01 ) -
        OUTDATASET(CICSTS31.CICS.DFHDPFMB)
//SYS01 DD *
    DDUMMY RECORD !! DO NOT ALTER !!
    EEXAMPLE RECORD REMOVE THIS LINE IF SAMPLES NOT REQUIRED
/*
//DEFAULT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    DEFINE ALTERNATEINDEX -
        ( NAME(CICSTS31.CICS.DFHDPFMX ) -
        RECORDS(1000) -
        VOLUME(&DSVOL) -
        KEYS(12 20) -
        RELATE(CICSTS31.CICS.DFHDPFMB) -
        RECORDSIZE(200 200) -
        SHAREOPTIONS(2 3) -
        UPGRADE ) -
    DATA -
        ( NAME(CICSTS31.CICS.DFHDPFMX.DATA) ) -
    INDEX -
        ( NAME(CICSTS31.CICS.DFHDPFMX.INDEX) )
    DEFINE PATH -
        ( NAME(CICSTS31.CICS.DFHDPFMP) -
        PATHENTRY(CICSTS31.CICS.DFHDPFMX) )
/*
//BLDDP EXEC PGM=IDCAMS
//BDSET1 DD DSN=CICSTS31.CICS.DFHDPFMB,DISP=SHR
//ADSET1 DD DSN=CICSTS31.CICS.DFHDPFMX,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    BLDINDEX -
        INFILE(BDSET1) -
        OUTFILE(ADSET1)
/*
//*/

```

Figure 29. Sample JCL to create the debugging profiles data sets

The sample JCL creates data sets which contain example debugging profiles. To create empty data sets, remove the following line:

```
EXAMPLE RECORD REMOVE THIS LINE IF SAMPLES NOT REQUIRED
```

Alternatively, you can run the CICS-supplied job DFHDEFDS (in CICSTS31.XDFHINST), to create the data sets for a CICS region. For information about the DFHDEFDS job, see the *CICS Transaction Server for z/OS Installation Guide*.

---

## Defining the debugging profiles data sets as VSAM RLS files

Define VSAM RLS files when you want to share profiles between several CICS regions in the same sysplex. CICS provides the following sample definitions:

```
*-----*
*   Define base file for Debugging Profiles (RLS)   *
*-----*
DEFINE FILE(DFHDPFMB) GROUP(DFHDPVR)
DESCRIPTION(Debugging Profiles base file - VSAM RLS)
      RLSACCESS(YES)          TABLE(NO)
      LSRPOOLID(1)           DSNSHARING(ALLREQS)
      STRINGS(5)             STATUS(ENABLED)
      OPENTIME(FIRSTREF)     DISPOSITION(SHARE)
      DATABUFFERS(3)        INDEXBUFFERS(2)
      RECORDFORMAT(V)       READINTEG(REPEATABLE)
      ADD(YES)               BROWSE(NO)
      DELETE(YES)           READ(YES)
      UPDATE(YES)           JOURNAL(NO)
      JNLREAD(NONE)         JNLSYNCREAD(NO)
      JNLUPDATE(NO)        JNLADD(NONE)
      JNLSYNCWRITE(YES)     RECOVERY(NONE)
      FWDRECOVLOG(NO)       BACKUPTYPE(STATIC)
*-----*
*   Define path file for Debugging Profiles (RLS)  *
*-----*
DEFINE FILE(DFHDPFMP) GROUP(DFHDPVR)
DESCRIPTION(Debugging Profiles path file - VSAM RLS)
      RLSACCESS(YES)          TABLE(NO)
      LSRPOOLID(1)           DSNSHARING(ALLREQS)
      STRINGS(10)            STATUS(ENABLED)
      OPENTIME(FIRSTREF)     DISPOSITION(SHARE)
      DATABUFFERS(11)       INDEXBUFFERS(10)
      RECORDFORMAT(V)       READINTEG(REPEATABLE)
      ADD(YES)               BROWSE(NO)
      DELETE(YES)           READ(YES)
      UPDATE(YES)           JOURNAL(NO)
      JNLREAD(NONE)         JNLSYNCREAD(NO)
      JNLUPDATE(NO)        JNLADD(NONE)
      JNLSYNCWRITE(YES)     RECOVERY(NONE)
      FWDRECOVLOG(NO)       BACKUPTYPE(STATIC)
```

Figure 30. Resource definitions for debugging profiles data sets defined as VSAM RLS files

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNAME attribute:
  - For DFHDPFMB, specify the name of the debugging profiles base data set. For example: DSNAME(CICSTS31.CICS.DFHDPFMB)
  - For DFHDPFMP, specify the name of the debugging profiles base data set. For example: DSNAME(CICSTS31.CICS.DFHDPFMP)

Alternatively, you can omit the DSNNAME attribute, and include a DD card in the CICS startup JCL. For example:

```
//DFHDPFMB DD DSN=CICSTS31.CICS.DFHDPFMB,DISP=SHR
//DFHDPFMP DD DSN=CICSTS31.CICS.DFHDPFMP,DISP=SHR
```

3. Install the FILE definitions.

---

## Defining the debugging profiles data sets as VSAM non-RLS files

Define VSAM non-RLS files when you do not need to share profiles between regions in the same sysplex. CICS provides the following sample definitions:

```
*-----*
*   Define base file for Debugging Profiles (non-RLS)   *
*-----*
DEFINE FILE(DFHDPFMB) GROUP(DFHDP)
DESCRIPTION(Debugging Profiles Base File)
    RLSACCESS(NO)          LSRPOOLID(1)
    READINTEG(UNCOMMITTED) DSNSHARING(ALLREQS)
    STRINGS(10)           STATUS(ENABLED)
    OPENTIME(FIRSTREF)    DISPOSITION(SHARE)
    DATABUFFERS(11)       INDEXBUFFERS(10)
    TABLE(NO)            RECORDFORMAT(V)
    ADD(YES)              BROWSE(YES)
    DELETE(YES)           READ(YES)
    UPDATE(YES)           JOURNAL(NO)
    JNLREAD(NONE)        JNLSYNCREAD(NO)
    JNLUPDATE(NO)        JNLADD(NONE)
    JNLSYNCWRITE(NO)     RECOVERY(NONE)
    FWDRECOVLOG(NO)      BACKUPTYPE(STATIC)
*-----*
*   Define path file for Debugging Profiles (non-RLS)   *
*-----*
DEFINE FILE(DFHDPFMP) GROUP(DFHDPVSL)
DESCRIPTION(Debugging Profiles Path File)
    RLSACCESS(NO)          LSRPOOLID(1)
    READINTEG(UNCOMMITTED) DSNSHARING(ALLREQS)
    STRINGS(10)           STATUS(ENABLED)
    OPENTIME(FIRSTREF)    DISPOSITION(SHARE)
    DATABUFFERS(11)       INDEXBUFFERS(10)
    TABLE(NO)            RECORDFORMAT(V)
    ADD(YES)              BROWSE(YES)
    DELETE(YES)           READ(YES)
    UPDATE(YES)           JOURNAL(NO)
    JNLREAD(NONE)        JNLSYNCREAD(NO)
    JNLUPDATE(NO)        JNLADD(NONE)
    JNLSYNCWRITE(NO)     RECOVERY(NONE)
    FWDRECOVLOG(NO)      BACKUPTYPE(STATIC)
```

*Figure 31. Resource definitions for debugging profiles data sets defined as VSAM non-RLS files*

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNNAME attribute:
  - For DFHDPFMB, specify the name of the debugging profiles base data set. For example: DSNNAME(CICSTS31.CICS.DFHDPFMB)
  - For DFHDPFMP, specify the name of the debugging profiles base data set. For example: DSNNAME(CICSTS31.CICS.DFHDPFMP)

Alternatively, you can omit the DSNNAME attribute, and include a DD card in the CICS startup JCL. For example:

```
//DFHDPFMB DD DSN=CICSTS31.CICS.DFHDPFMB,DISP=SHR
//DFHDPFMP DD DSN=CICSTS31.CICS.DFHDPFMP,DISP=SHR
```

3. Install the FILE definitions.

---

## Defining the debugging profiles data sets as remote files

Define remote files when you want to use profiles that are stored in a region that is connected using MRO or ISC. CICS provides the following sample definitions:

```
*-----*
*   Define base file for Debugging Profiles (non-RLS remote)   *
*-----*
DEFINE FILE(DFHDPFMB) GROUP(DFHDPVSR)
DESCRIPTION(Debugging Profile Base File - VSAM Remote)
          REMOTESYSTEM(CICA)          REMOTENAME(DFHDPFMB)
*-----*
*   Define path file for Debugging Profiles (non-RLS remote)   *
*-----*
DEFINE FILE(DFHDPFMP) GROUP(DFHDPVSR)
DESCRIPTION(Debugging Profile Path File - VSAM Remote)
          REMOTESYSTEM(CICA)          REMOTENAME(DFHDPFMP)
```

*Figure 32. Resource definitions for debugging profiles data sets defined as remote files*

1. Copy the sample FILE definitions for DFHDPFMB and DFHDPFMP resource to another group.
2. Add the DSNAME attribute:
  - For DFHDPFMB, specify the name of the debugging profiles base data set. For example: DSNAME(CICSTS31.CICS.DFHDPFMB)
  - For DFHDPFMP, specify the name of the debugging profiles base data set. For example: DSNAME(CICSTS31.CICS.DFHDPFMP)

Alternatively, you can omit the DSNAME attribute, and include a DD card in the CICS startup JCL. For example:

```
//DFHDPFMB DD DSN=CICSTS31.CICS.DFHDPFMB,DISP=SHR
//DFHDPFMP DD DSN=CICSTS31.CICS.DFHDPFMP,DISP=SHR
```

3. Install the FILE definitions.

If you define remote files, you will need to install corresponding file definitions in the remote system.



---

## Part 3. CICS system initialization

This section of the book describes how to define CICS system initialization parameters and how they are processed by CICS. It also describes a startup job stream you can use to start a CICS system.

- Chapter 18, “Specifying CICS system initialization parameters,” on page 151 describes the CICS system initialization parameters.
- Chapter 19, “Processing system initialization parameters,” on page 273 describes the use of the PARM parameter, the SYSIN data set, and the system console for supplying system initialization parameters, and how these are processed by CICS.
- Chapter 21, “CICS startup,” on page 337 describes a sample startup job stream, and a sample procedure for use as a started task.



---

## Chapter 18. Specifying CICS system initialization parameters

This chapter describes the CICS system initialization parameters, which you can use to modify CICS system attributes when you start your CICS regions. It gives the syntax and a detailed description of each system initialization parameter, and describes the methods that you can use to define the parameters to CICS.

The primary method of providing system initialization parameters is with a system initialization table (SIT). The parameters of the SIT, which you assemble as a load table, supply the system initialization program with most of the information necessary to initialize the system to suit your unique environment. You can generate more than one SIT, and at the time of system initialization select the one that is appropriate to your needs.

You can also specify other system initialization parameters, which cannot be coded in the SIT. You specify which SIT you want, and other system initialization parameters (with a few exceptions), in any of three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator's console

You can also use these methods of input to the system initialization process to override most of the system initialization parameters assembled in the SIT.

The information defined by system initialization parameters can be grouped into three categories:

1. Information used to initialize and control CICS system functions (for example, information such as the dynamic storage area limits and the region exit time interval)
2. Module suffixes used to load your own versions of CICS control tables (for example, DFHMCTxx)
3. Special information used to control the initialization process

The syntax of the system initialization parameters that can be coded in the DFHSIT macro is listed in Table 17 on page 157. Except for those parameters marked “**SIT macro only**”, all the system initialization parameters can be provided at run time, although there are restrictions in some cases. The restrictions are explained at the end of the description of the system initialization parameter to which they apply. See the CHKSTRM parameter on page “CHKSTRM” on page 171 for an example of such a restriction.

There are some other CICS system initialization parameters (and options of the parameters in Table 17 on page 157) that you cannot define in the DFHSIT macro. (See “Initialization parameters that cannot be coded in the DFHSIT macro” on page 161.) The parameters that you cannot define in the DFHSIT macro are shown in Figure 33 on page 162.

### Migration considerations

If you have existing system initialization tables, you must modify them. Remove all obsolete parameters, and specify the required values for new or changed parameters if you want to run with other than the defaults. When you have made the necessary changes, reassemble the tables using the CICS Transaction Server for z/OS, Version 3 Release 1 macro libraries.

If you have system initialization parameters defined in CICS start-up procedures, you must modify these also.

To avoid generating specific system initialization tables for each CICS region, a simple solution is to let CICS load the default, unsuffixed table (DFHSIT) at start-up, and supply the system initialization parameters for each region in a SYSIN data set. For more information about the source of the default system initialization table, see “The default system initialization table” on page 263.

This chapter describes:

- “Specifying DFHSIT macro parameters” on page 157
- “The default system initialization table” on page 263
- “The system initialization parameter descriptions” on page 164
- “Assembling the SIT” on page 270
- “Selecting versions of CICS programs and tables” on page 270

---

## System initialization parameters for open TCBs

The open transaction environment (OTE) function was added to CICS Transaction Server for OS/390, Version 1 Release 3 and later versions.

In earlier releases of CICS, CICS ran all user transactions under a single z/OS TCB, the CICS quasi-reentrant (QR) TCB. Direct invocation of other services outside the scope of the CICS permitted interfaces could interfere with the use by CICS of the QR TCB. In particular, requests that resulted in the suspension (“blocking”) of the QR TCB, which happens when an MVS wait is issued, would cause all CICS tasks to wait. For example, some services provided by DB2, MVS, UNIX System Services, or TCP/IP, might result in TCB blocking.

The open transaction environment is an environment where CICS application code can use non-CICS services (facilities outside the scope of the CICS API) within the CICS address space, without interference with other transactions. Applications that exploit the open transaction environment run on their own open TCB, rather than on the QR TCB. Unlike the QR TCB, CICS does not perform sub-dispatching on an open TCB. If the application running on an open TCB invokes a non-CICS service that blocks the TCB, the TCB blocking does not affect other CICS tasks.

There are several open TCB *modes*, each given 2-character identifiers. Each mode has a specific purpose, and is handled by CICS in a different way:

### J8 mode TCBs and J9 mode TCBs

are both used to run Java programs under a Java Virtual Machine (JVM). The JVM is created on the TCB. J8 TCBs are used for JVMs in CICS key, and J9 mode TCBs are used for JVMs in user key. “How CICS manages JVMs in the JVM pool” in *Java Applications in CICS* has more information about how CICS manages JVMs and their TCBs.

### L8 mode TCBs and L9 mode TCBs

are both used to run OPENAPI programs, that is those defined as OPENAPI by their PROGRAM resource definition.

- L8 mode TCBs are used for CICSKEY OPENAPI application programs.
- L9 mode TCBs are used for USERKEY OPENAPI application programs.

### L8 mode TCBs

are also used when programs need access to a resource manager through a task-related user exit (TRUE) enabled using the OPENAPI option on the ENABLE PROGRAM command.

When CICS is connected to DB2, the CICS DB2 task-related user exit operates in OPENAPI mode (it is an open API TRUE). In this situation, the CICS DB2 attachment facility uses L8 TCBs for DB2 request processing. (Prior to this, the CICS DB2 attachment facility had to create and manage its own subtask thread TCBs with which to access DB2 resources, so that waits for DB2 resources would not block the QR TCB.) “Overview: How threads work” in the *CICS DB2 Guide* has more information about how CICS uses open TCBs as thread TCBs for the CICS DB2 attachment facility. “Enabling CICS DB2 applications to exploit the open transaction environment (OTE) through threadsafe programming” in the *CICS DB2 Guide* explains what your CICS DB2 application programs must do in order to gain performance benefits by continuing to run on the L8 mode TCB after the DB2 request has been completed.

### **L8 mode TCBs**

are also used by CICS itself, because CICS uses OPENAPI CICSKEY programs which run on L8 TCBs:

- when accessing doctemplates and HTTP static responses that are stored on Hierarchical File System (HFS).
- when processing WebService requests and parsing XML.

### **SP mode TCB and S8 mode TCBs**

are used by CICS to manage SSL connections. The S8 TCBs run within a single enclave, which is owned by the SP TCB and also contains the SSL cache. An S8 TCB is allocated to a task from the SSL pool, but is only locked for the period of time it takes to perform SSL functions such as the SSL handshake. After the SSL negotiation is complete, the TCB is released back into the SSL pool to be reused.

In UNIX System Services, the MAXTHREADS and MAXTHREADTASKS parameters can be used to restrict the number of pthreads that a USS process can own. Each SSL TCB requires a pthread and an MVS task. You should therefore ensure that the values of these USS parameters exceed the MAXSSLTCBS system initialization parameter. If you do not set a large enough value for MAXTHREADS or MAXTHREADTASKS and CICS reaches one of these limits while attempting to attach an SSL TCB, CICS issues error message DFHDS0002 severe error code X'0137' from DFHDSIT.

### **X8 mode TCBs and X9 mode TCBs**

are both used to run C and C++ programs compiled with the XPLINK option. X8 TCBs are used for programs in CICS key, and X9 mode TCBs are used for programs in user key. Each instance of an XPLink program uses one X8 or X9 TCB. in the *CICS Application Programming Guide* has more information about using XPLink.

A CICS task is allowed as many J8, J9, X8 and X9 TCBs as it requires, and these TCBs are only kept until program termination, but each CICS task is allowed at most one L8 and one L9 TCB, and it keeps an L8 and an L9 TCB from the time it is allocated to the end of the task. The TCBs then become free, and CICS can allocate them to another task, or destroy them.

CICS manages open TCBs in *pools*. A pool contains open TCBs that are used for the same purpose; for example, there is a pool of L8- and L9-mode open TCBs and a pool of J8- and J9-mode open TCBs (which is known as the JVM pool). The priority of the open TCBs in the JVM pool is set lower than that of the main CICS

QR TCB, to ensure that the J8 and J9 TCB activity does not affect the main CICS workload that is being processed on the CICS QR TCB.

The maximum number of TCBs allowed in each pool is specified by a MAXxxxxTCBS parameter:

- The “MAXOPENTCBS” on page 210 parameter limits the number of TCBs in the pool of L8 and L9 mode open TCBs.
- The “MAXJVMTCBS” on page 210 parameter limits the number of TCBs in the pool of J8- and J9-mode open TCBs. (It applies to the maximum total number of J8- and J9-mode TCBs in the JVM pool, and CICS decides how many of them should be J8 TCBs and how many should be J9 TCBs, according to the number of requests that specify each execution key.)
- The “MAXSSLTCBS” on page 211 parameter limits the number of TCBs in the pool of S8 mode open TCBs.
- The “MAXXPTCBS” on page 211 parameter limits the number of TCBs in the pool of X8 and X9 mode open TCBs.

The minimum permitted value for any of the MAXxxxxTCBS parameters is 1, meaning that CICS is always able to create at least 1 open TCB in each mode. CICS can create (or attach) open TCBs in each pool up to the limit set by the corresponding MAXxxxxTCBS parameter.

At any one time, a pool can consist of some TCBs that are allocated to tasks, and others that have been freed by applications and are available for reuse. For example, if the maximum number of open TCBs for JVMs (J8- or J9-mode) is set to 10, at a particular time the pool could consist of 5 TCBs, not all of which are allocated.

When an application makes a request that involves the use of an open TCB (for example, a request to execute a Java program in a JVM), CICS first tries to find a suitable TCB that is available for reuse in the appropriate pool of open TCBs. CICS can match a request with an available TCB of the correct mode only if the TCB has matching attributes. For example, in the case of a request for a J8- or J9-mode TCB, a free JVM TCB can be allocated only if the JVM profile names also match.

CICS attaches a new TCB if it can't find a suitable match with a free TCB, provided that the MAXxxxxTCBS limit for the pool has not been reached. For the JVM pool, where the creation of a JVM can involve a large amount of MVS storage, an additional safeguard is provided by the CICS storage monitor for MVS storage, which prevents the creation of new JVMs if MVS storage is severely constrained.

If CICS cannot find a suitable match, and the MAXxxxxTCBS limit for the pool has been reached, CICS might fulfil the request by destroying a free TCB that has the wrong attributes, and replacing it with a TCB that has the right attributes. This technique is called stealing. Stealing can be costly on performance, depending on the type of open TCB, so CICS avoids it where it makes sense to do so. For L8 mode TCBs (used by task-related user exits enabled with OPENAPI), the cost of stealing a TCB is low, so CICS always steals a TCB if it receives a request for which it cannot find a suitable match with a free TCB or attach a new TCB. However, for J8- and J9-mode (JVM) TCBs, the cost of TCB stealing is high, because CICS needs to destroy and re-initialize the JVM as well as the TCB; so CICS has a selection mechanism to decide if stealing the TCB is worthwhile, or if the request should be made to wait. CICS maintains statistics of excess TCB management and TCB stealing activities.

To minimize the impact on storage, CICS attempts to balance the number of open TCBs in each pool against current needs by reducing the number of free TCBs. If CICS finds that there are free TCBs in a pool, it gradually removes the excess number by detaching them, thereby freeing the resources used by the excess TCBs.

When specifying any of the MAXxxxxTCBS parameters, take into account TCB storage requirements. All TCBs use real storage, and virtual storage below 16MB, so the number of open TCBs that a CICS region can support is restricted by the amount of storage available both above and below 16MB. JVMs, which run on J8- and J9-mode TCBs, use a large amount of storage above 16MB in addition to the cost of the TCB. “Managing your JVM pool for performance” in the *CICS Performance Guide* has guidance for calculating the number of JVMs that your CICS region can support.

---

## # TCB considerations with UNIX System Services

# When defining the numbers of TCBs that are allowed in a CICS region, you need to  
# consider the settings in UNIX System Services (USS) that control the number of  
# processes that can run within a CICS region.

# In USS, the MAXPROCUSER parameter specifies the maximum number of  
# processes one USS UNIX user identifier (UID) can have concurrently active,  
# regardless of how the processes were created. The value can be in the range 3 to  
# 32767. The default is 25. The MAXPROCUSER parameter is specified in  
# SYS1.PARMLIB member BPXPRMxx. The *z/OS MVS Initialization and Tuning*  
# *Reference* gives guidance on the setting of MAXPROCUSER.

# MAXPROCUSER is independent of any particular userid. However, there is an  
# equivalent RACF setting to limit the number of processes by userid for a particular  
# user. This is PROCUSERMAX. It sets the maximum number of processes per  
# userid field of the RACF OMVS SEGMENT of a userid's profile.

# The following TCBs all contribute towards the potential number of processes  
# associated with a particular CICS region.

- # • MAXOPENTCBS - the maximum number of L8 TCBs that can exist.
- # • MAXJVMTCBS - the maximum number of J8 and J9 TCBs that can exist
- # • MAXSSLTCBS - the maximum number of S8 TCBs that can exist.
- # • MAXXPTCBS - the maximum number of X8 and X9 TCBs that exist.
- # • The SO TCB - used to issue the necessary USS and CEEPIPI calls for the  
# Socket domain.
- # • The SL TCB - provides a listening environment for Sockets domain requests.
- # • The SP TCB - owns the S8 TCBs and the SSL cache.
- # • TCBs used by the separate “TCP/IP Socket Interface for CICS” component of the  
# z/OS Communications Server (if applicable).

# By adding the number of TCBs from the above list, it is possible to obtain the total  
# number of processes that might be associated with a given CICS region. This total  
# represents a possible upper limit for the region.

# Where you have CICS systems that share the same userid, add the totals together  
# to give the maximum number of processes associated with that userid. This is  
# because MAXPROCUSER is the number of processes for a UID, not for each job.

# When you have determined the total possible number of processes associated with  
# each userid for your CICS regions, use the largest number and add an extra 10%  
# to this figure when calculating the value of MAXPROCUSER.

# If you have a particular userid with a high result for the total number of processes  
# required, due to several CICS systems sharing the same userid, setting  
# MAXPROCUSER to such a figure might not be appropriate. In this situation, use  
# the PROCUSERMAX parameter on the OMVS segment of the RACF profile for the  
# userid to set a suitably high value to accommodate the requirements of the userid.

# The setting of the MAXPROCUSER and PROCUSERMAX parameters does not in  
# itself consume extra resources. These are limiting values. CICS does not generate  
# the open TCBS until they are needed, meaning that processes and system  
# resources are not associated with TCBS until required. Note that TCBS specified in  
# the SSLTCBS system initialization parameter are created at CICS system  
# initialization.

# The setting of the TCPIP system initialization parameter does not affect the use of  
# open TCBS by the OTE. Also, if you specify TCPIP=NO and no OTE-managed  
# services are used by CICS, then two of the MAXPROCUSER entries will be used in  
# the initialization of the Sockets domain.

## # **The implications of setting MAXPROCUSER too low**

# If you do not set a large enough value for MAXPROCUSER for the CICS  
# environment, you might get a number of warning and error messages. These are  
# described below:

### # **Message BPXI040I**

# A USS warning message that alerts the operator that system resources are  
# being consumed. The message notifies the operator when a threshold of  
# 85% of the MAXPROCUSER value for a given UNIX Process Identifier  
# (PID) has been reached. It is possible for the percentage to exceed 100%.  
# This is because two special UIDs are allowed to create more processes  
# than MAXPROCUSER would normally allow. The superuser (UID=0) can  
# exceed many of the limits set in BPXPRMxx. Also, the default-UID can  
# exceed the MAXPROCUSER setting. This is because many users can  
# make use of the default-UID, and they each have independent processes. If  
# each user were given an individual UID, then each would be subject to  
# MAXPROCUSER independently. The default-UID refers to a RACF userid  
# without an OMVS segment defined for it; as such, it uses the default OMVS  
# segment. The default-UID should not to be confused with the CICS default  
# user.

### # **Message DFHKE0500**

# This message is issued by the CICS TS for z/OS Version 3.1 Kernel when  
# MAXPROCUSER has been exceeded for the userid of the CICS system.  
# This could occur because a number of CICS systems are sharing the same  
# userid on USS and have a requirement to use a number of TCBS that is  
# greater than the value defined in the MAXPROCUSER parameter.

## Specifying DFHSIT macro parameters

# You should code the DFHSIT parameters and keywords in uppercase, except for  
 # parameters where case is important. For example, any parameter that specifies the  
 # name of an HFS directory is case sensitive.

Table 17. The DFHSIT macro parameters

#	DFHSIT [TYPE={ <b>CSECT</b>  DSECT}] [,ADL={ <b>30</b>  number}] [,AIBRIDGE={ <b>AUTO</b>  YES}] [,AICONS={ <b>NO</b>  YES AUTO}] [,AIEEXIT={ <b>DFHZATDX</b>  DFHZATDY name}] [,AILDELAY={ <b>0</b>  hhmmss}] [,AIQMAX={ <b>100</b>  number}] [,AIRDELAY={ <b>700</b>  hhmmss}] [,AKPFREQ={ <b>4000</b>  number}] [,APPLID={({ <b>DBDCCICS</b>  name1},{name2})}] [,AUTCONN={ <b>0</b>  hhmmss}] [,AUTODST={ <b>NO</b>  YES}] [,AUTORESETTIME={ <b>NO</b>  YES}] [,AUXTR={ <b>OFF</b>  ON}] [,AUXTRSW={ <b>NO</b>  ALL NEXT}] [,BMS={({MINIMUM STANDARD  <b>FULL</b> },{COLD}[ [, <b>UNALIGN</b>  ALIGN]},{ <b>DDS</b>  NODDS})}] [,BRMAXKEEPTIME={ <b>86400</b>  number}] [,CDSASZE={ <b>OK</b>  number}] [,CICSSVC={ <b>216</b>  number}] [,CILOCK={ <b>NO</b>  YES}] [,CLSDSTP={ <b>NOTIFY</b>  NONOTIFY}] [,CLT=xx] [,CMDPROT={ <b>YES</b>  NO}] [,CMDSEC={ <b>ASIS</b>  ALWAYS}] [,CONFDATA={ <b>SHOW</b>  HIDETC}] [,CONFTXT={ <b>NO</b>  YES}] [CRLPROFILE= <i>PROFILENAME</i> ] [,CSDACC={ <b>READWRITE</b>  READONLY}] [,CSDBKUP={ <b>STATIC</b>  DYNAMIC}] [,CSDBUFND=number] [,CSDBUFNI=number] [,CSDDISP={OLD SHR}] [,CSDDSN={name}] [,CSDFRLOG=number] [,CSDINTEG={ <b>UNCOMMITTED</b>  CONSISTENT  REPEATABLE}] [,CSDJID={ <b>NO</b>  number}] [,CSDLRNO={ <b>1</b>  number NONE NO}] [,CSDRECOV={ <b>NONE</b>  ALL BACKOUTONLY}] [,CSDRLS={ <b>NO</b>  YES}] [,CSDSTRNO={ <b>2</b>  number}] [,CWAKEY={ <b>USER</b>  CICS}] [,DAE={ <b>NO</b>  YES}] [,DATFORM={ <b>MMDDYY</b>  DDMMYY YYMMDD}] [,DB2CONN={ <b>NO</b>  YES}] [,DBCTLCON={ <b>NO</b>  YES}] [,DEBUGTOOL={ <b>NO</b>  YES}]
#	
#	

Table 17. The DFHSIT macro parameters (continued)

	<p>[,DFLTUSER={<b>CICSUSER</b> userid}]</p> <p>[,DIP={<b>NO</b> YES}]</p> <p>[,DISMACP={<b>YES</b> NO}]</p> <p>[,DOCCODEPAGE={<b>037</b> codepage}]</p> <p>[,DSALIM={<b>5M</b> number}]</p> <p>[,DSHIPIDL={<b>020000</b> hhmmss}]</p> <p>[,DSHIPINT={<b>120000</b> hhmmss}]</p> <p>[,DSRTPGM={<b>NONE</b> DFHDSRP program-name EYU9XLOP}]</p> <p>[,DTRPGM={<b>DFHDYP</b> program-name}]</p> <p>[,DTRTRAN={<b>CRTX</b> name NO}]</p> <p>[,DUMP={<b>YES</b> NO}]</p> <p>[,DUMPDS={<b>AUTO</b> AIB}]</p> <p>[,DUMPSW={<b>NO</b> NEXT}]</p> <p>[,DURETRY={<b>30</b> number-of-seconds 0}]</p> <p>[,ECDSASZE={<b>OK</b> number}]</p> <p>[,EDSALIM={<b>30M</b> number}]</p> <p>[,EJBROLEPRFX=<i>ejbrole-prefix</i>]</p> <p>[,ENCRYPTION={WEAK MEDIUM <b>STRONG</b>}]</p> <p>[,EODI={<b>E0</b> xx}]</p> <p>[,ERDSASZE={<b>OK</b> number}]</p> <p>[,ESDSASZE={<b>OK</b> number}]</p> <p>[,ESMEXITS={<b>NO</b> INSTLN INSTLN}] (<i>SIT macro only</i>)</p> <p>[,EUDSASZE={<b>OK</b> number}]</p> <p>[,FCT={<b>NO</b> xx YES}]</p> <p>[,FEPI={<b>NO</b> YES}]</p> <p>[,FLDSEP={'_' 'xxx'}]</p> <p>[,FLDSTRT={'_' 'x'}]</p> <p>[,FORCEQR={<b>NO</b> YES}]</p> <p>[,FSSTAFF={YES <b>NO</b>}]</p> <p>[,FTIMEOUT={<b>30</b> nn}]</p> <p>[,GMTEXT={'<b>WELCOME TO CICS</b>' 'text'}]</p> <p>[,GMTRAN={<b>CSGM</b> CESN name}]</p> <p>[,GNTRAN={<b>NO</b> transaction-id}]</p> <p>[,GRNAME=name]</p> <p>[,GRPLIST={<b>DFHLIST</b> name (name[,name2][,name3][,name4])}]</p> <p>[,GTFTR={<b>OFF</b> ON}]</p> <p>[,HPO={<b>NO</b> YES}] (<i>SIT macro only</i>)</p> <p>[,ICP=COLD]</p> <p>[,ICV={<b>1000</b> number}]</p> <p>[,ICVR={<b>5000</b> number}]</p> <p>[,ICVTS=500 number}]</p> <p>[,IIOPLISTENER={<b>YES</b> NO}]</p> <p>[,INFOCENTER={'<i>infocenter_urf</i>'}]</p> <p>[,INITPARM=(pgmname_1='parmstring_1' [, ...,pgmname_n='parmstring_n'])]</p> <p>[,INTTR={<b>ON</b> OFF}]</p> <p>[,IRCSTRT={<b>NO</b> YES}]</p> <p>[,ISC={<b>NO</b> YES}]</p> <p>[,JESDI={<b>30</b> number}]</p> <p>[,JVMCCPROFILE={<b>DFHJVMCC</b> profile}]</p> <p>[,JVMCCSIZE={<b>24M</b> number}]</p> <p>[,JVMCCSTART={<b>AUTO</b> YES NO}]</p> <p>[,JVMPROFILEDIR={/<i>usr/lpp/cicsts/cicsts31/JVMProfiles</i> directory}]</p> <p>[,KEYRING={'<i>key-database-path-name</i>'}]</p>
--	--

Table 17. The DFHSIT macro parameters (continued)

		<pre>[,LGDFINT={30 number}] [,LGNMSG={NO YES}] [,LLACOPY={YES NO NEWCOPY}] [,LOCALCCSID={037 CCSID}] [,LPA={NO YES}] [,MAXJVMTCBS={5 number}] [,MAXOPENTCBS={12 number}] [,MAXSOCKETS=number] [,MAXSSLTCBS={8 number}] [,MAXXPTCBS={5 number}] [,MCT={NO YES xx}] [,MN={OFF ON}] [,MNCONV={NO YES}] [,MNEVE={OFF ON}] [,MNEXC={OFF ON}] [,MNFREQ={0 hhmmss}] [,MNPER={OFF ON}] [,MNPRES={OFF ON}] [,MNSUBSYS={null xxxx}] [,MNSYNC={NO YES}] [,MNTIME={GMT LOCAL}] [,MQCONN={NO YES}] [,MROBTCH={1 number}] [,MROFSE={NO YES}] [,MROLRM={NO YES}] [,MSGCASE={MIXED UPPER}] [,MSGLVL={1 0}] [,MXT={5 number}] [,NATLANG=(E,x,y,z,...)] [,NCPLDFT={DFHNC001 name}] [,OPERTIM={120 number}] [,OPNDLIM={10 number}] [,PARMERR={INTERACT IGNORE ABEND}] [,PDI={30 number}] [,PDIR={NO YES xx}] [,PGAICTLG={MODIFY NONE ALL}] [,PGAEXIT={DFHPGADX name}] [,PGAIPGM={INACTIVE ACTIVE}] [,PGCHAIN=character(s)] [,PGCOPY=character(s)] [,PGPURGE=character(s)] [,PGRET=character(s)] [,PLTPI={NO xx YES}] [,PLTPISEC={NONE CMDSEC RESSECIALL}] [,PLTPIUSR=userid] [,PLTSD={NO xx YES}] [,PRGDLAY={0 hhmm}] [,PRINT={NO YES PA1 PA2 PA3}] [,PRTYAGE={32768 number}] [,PSBCHK={NO YES}] [,PSDINT={0 hhmmss}] [,PSTYPE={SNPS MNPS}] [,PVDELAY={30 number}] [,QUIESTIM={240 number}]</pre>
--	--	--

Table 17. The DFHSIT macro parameters (continued)

	<pre> [.RAMAX={256 number}] [.RAPOOL={ 50 value1 (value1,value2,FORCE)}] [.RDSASZE={OK number}] [.RENTPGM={PROTECT NOPROTECT}] [.RESP={FME IRRN}] [.RESSEC={ASIS ALWAYS}] [.RLS={NO YES}] [.RLSTOLSR={NO YES}] [.RMTRAN={({GMTRAN-name name1}             [, {GMTRAN-name name2}})] [.RRMS={NO YES}] [.RST={NO xx YES}] [.RSTSIGNOFF={NOFORCE FORCE}] [.RSTSIGNTIME={500 time}] [.RUWAPOOL={NO YES}] [.SDSASZE={OK number}] [.SDTRAN={CESD name_of_shutdown_tran NO}] [.SEC={YES NO}] [.SECPREFIX={NO YES prefix}] [.SKRxxxx='page-retrieval-command'] [.SNSCOPE={NONE CICSIMVMSIMAGE SYSPLEX}] [.SPCTR={1,2 1[,2][,3] ALL OFF}] [.SPOOL={NO YES}] [.SRBSVC={215 number}] [.SRT={1 YES xx NO}] [.SSLDELAY={600 number}] [.SSLTCBS={8 number}] [.START={AUTO INITIAL COLD STANDBY}] [.STARTER={NO YES}] (<i>SIT macro only</i>) [.STATRCD={OFF ON}] [.STGPROT={NO YES}] [.STGRCVY={NO YES}] [.STNTR={1 (1[,2][,3]) ALL OFF}] [.SUBTSKS={0 1}] [.SUFFIX=xx] (<i>SIT macro only</i>) [.SYDUMAX={999 number}] [.SYSIDNT={CICS name}] [.SYSTR={ON OFF}] [.TAKEOVR={MANUAL AUTO COMMAND}] [.TBEXITS=( name1 [,name2 [,name3             [,name4 [,name5 [,name6 )] [.TCP={YES NO}] [.TCPIP={NO YES}] [.TCSACTN={NONE UNBIND FORCE}] [.TCSWAIT={4 number NO NONE 0}] [.TCT={NO xx YES}] [.TCTUAKEY={USER CICS}] [.TCTUALOC={BELOW ANY}] [.TD={({3 number1} [, {3 number2}})] [.TDINTRA={NOEMPTY EMPTY}] [.TRANISO={NO YES}] [.TRAP={OFF ON}] [.TRDUMAX={999 number}] [.TRTABSZ={16 number}] </pre>
--	---

Table 17. The DFHSIT macro parameters (continued)

	<pre>[,TRTRANSZ={16 number}] [,TRTRANZY={TRAN ALL}] [,TS={COLD}[,{0 3 value-1}][,{3 value-2}]] [,TST={NO YES xx}] [,UDSASZE={OK number}] [,UOWNETQL=user_defined_value] [,USERTR={ON OFF}] [,USRDELAY={30 number}] [,VTAM={YES NO}] [,VTPREFIX={\ character}] [,WEBDELAY={5 time_out,60 keep_time}] [,WRKAREA={512 number}] [,XAPPC={NO YES}] [,XCMD={YES name NO}] [,XDB2={NO name}] [,XDCT={YES name NO}] [,XEJB={YES NO}] [,XFCT={YES name NO}] [,XJCT={YES name NO}] [,XLT={NO xx YES}] [,XPCT={YES name NO}] [,XPPT={YES name NO}] [,XPSB={YES name NO}] [,XRF={NO YES}] [,XTRAN={YES name NO}] [,XTST={YES name NO}] [,XUSER={YES NO}]</pre> <p>You must terminate your macro parameters with the following END statement.</p>
END	DFHSITBA

## Initialization parameters that cannot be coded in the DFHSIT macro

There are some CICS system initialization parameters that you cannot define in the DFHSIT macro. These parameters can be supplied only at CICS startup time in any of these three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator's console

The system initialization parameters that you cannot define in the DFHSIT macro are shown in Figure 33 on page 162.

For information about coding system initialization parameters in PARM, SYSIN, or at the console, see Chapter 19, "Processing system initialization parameters," on page 273.

```

CDSASZE={0K|number}
CHKSTRM={CURRENT|NONE}
CHKSTSK={CURRENT|NONE}
ECDSASZE={0K|number}
ERDSASZE={0K|number}
ESDSASZE={0K|number}
EUDSASZE={0K|number}
JVMLEVEL0TRACE={ALL(EXCEPTION)|user override string}
JVMLEVEL1TRACE={ALL(ENTRY,EXIT)|user override string}
JVMLEVEL2TRACE={ALL|user override string}
JVMUSERTRACE={NONE|user override string}
NEWSIT={YES|NO}
OFFSITE={NO|YES}
PRVMOD={name|(name,name...name)}
RDSASZE={0K|number}
SDSASZE={0K|number}
SIT=xx
START=(option,ALL)
UDSASZE={0K|number}

```

Figure 33. System initialization parameters you cannot code in the DFHSIT macro

## Defining CICS resource table and module keywords

Table 18 shows the system initialization keywords for those CICS resources that:

- Have a suffix option, or
- Result in a dummy program or table if you code resource=NO, or
- You can COLD start individually

Table 18. Summary of resources with a suffix, a dummy load module, or a COLD option

DFHSIT keyword <b>1</b>	Default <b>2</b>	Suffix <b>3</b>	Dummy <b>4</b>	COLD start <b>5</b>
BMS	FULL	-	-	COLD
CLT	-	xx	-	-
DIP	NO	-	program	-
FCT	YES	xx	-	-
ICP	-	-	-	COLD
MCT	NO	xx	<b>6</b>	-
PDIR	NO	xx	-	-
PLTPI	NO	xx	-	-
PLTSD	NO	xx	-	-
RST	NO	xx	-	-
SRT	YES	xx	-	-
TCT	YES	xx	table	-
TS	-	-	-	COLD
TST	NO	xx	-	-
XLT	NO	xx	-	-

### Notes:

**1** When the DFHSIT keyword is specified with more than one value, these values must be enclosed within parentheses: for example, BMS=(FULL,COLD).

**2** The **Default** column indicates the default value for the keyword in the DFHSIT macro.

For keywords with the suffix option, if you code YES in the SIT, an unsuffixed version of the table or program is loaded. For example, TCT=YES results in a table called DFHTCT being loaded. You can also select an unsuffixed module or table at CICS startup by specifying **keyword=**, or **keyword=YES**. For example, if you code: FCT=, or FCT=YES

blanks are appended to DFHFCT, and these unsuffixed names are used during initialization.

The result of specifying **keyword=**, as a system initialization parameter through PARM, SYSIN, or CONSOLE is not necessarily the same as in the DFHSIT macro. For example, TST=, (or omitted altogether) when coding the DFHSIT macro is taken to mean TST=NO, but TST=, through any of the other three methods is the same as TST=YES.

**3** The **Suffix** column indicates whether you can code a suffix. (xx indicates that a suffix can be coded.)

A suffix can be any 1 or 2 characters, but you must not use DY, and you cannot use NO as a suffix.

If you code a suffix, a table or program with that suffix appended to the standard name is loaded.

**4** The **Dummy** column indicates whether a dummy version of the program or table is loaded if you code NO. In some cases, coding NO for the operand associated with the **table** results in a dummy **program**. For more information about the effect of this option, see “Selecting versions of CICS programs and tables” on page 270.

**5** The **COLD start** column indicates whether the resource can be forced to start COLD. (COLD indicates that the resource can be cold started individually).

i1.TST and cold start Ensure that you cold start temporary storage or the whole system if you make any change to the TST.

If COLD is coded, it can be overridden only by coding START=(...,ALL) as a system initialization parameter. For more information about this option, see page “ALL” on page 240.

For more information about CICS table and program selection, see “Selecting versions of CICS programs and tables” on page 270.

**6** If you code MCT=NO, the CICS monitoring domain builds dynamically a default monitoring control table. This ensures that default monitoring control table entries are always available for use when monitoring is on and a monitoring class is active.

---

## The system initialization parameter descriptions

Unless otherwise stated, all of the system initialization parameters described here can be defined to CICS by any of these four ways:

1. In a DFHSIT macro
2. In a PARM parameter on the EXEC PGM=DFHSIP statement
3. In the SYSIN data set of the CICS startup job stream
4. Through the system console

Default values are **underscored**; for example, TYPE=CSECT. This notation applies to the SIT macro parameters only.

TYPE={CSECT|DSECT}

specifies the type of SIT to be generated.

CSECT A regular control section that is normally used.

DSECT A dummy control section.

ADI={30|number}

specifies the alternate delay interval in seconds for an alternate CICS region when you are running CICS with XRF. The minimum delay that you can specify is 5 seconds. This is the time that must elapse between the (apparent) loss of the surveillance signal in the active CICS region, and any reaction by the alternate CICS region. The corresponding parameter for the active is PDI. ADI and PDI need not have the same value.

**Note:** You must give careful consideration to the values you specify for the parameters ADI and JESDI so that they do not conflict with your installation's policy on PR/SM™ RESETTIME and the XCF INTERVAL and OPNOTIFY intervals. You should ensure that the sum of the interval you specify for ADI plus JESDI exceeds the interval specified by the XCF INTERVAL and the PR/SM policy interval RESETTIME.

AIBRIDGE={AUTO|YES}

specifies whether the autoinstall user replaceable module (URM) is to be called when creating bridge facilities (virtual terminals) used by the 3270 bridge mechanism.

AUTO This is the default, and specifies that bridge facilities are defined automatically by CICS. The autoinstall URM is not called.

**YES** Specifies that the autoinstall URM is to be called for all new bridge facilities.

See the *CICS Customization Guide* for information about writing an autoinstall user replaceable module.

AICONS={NO|YES|AUTO}

specifies whether you want autoinstall support for consoles. You can also set the state of autoinstall support for consoles dynamically using the CEMT, or EXEC CICS, SET AUTOINSTALL command.

NO This is the default, and specifies that the CICS regions does not support autoinstall for consoles.

**YES** Specifies that console autoinstall is active and CICS is to call the autoinstall control program, as part of the autoinstall process, when an undefined console issues an MVS MODIFY command to CICS.

**AUTO** Specifies that console autoinstall is active but CICS is not to call the autoinstall control program when an undefined console issues an MVS MODIFY command to CICS. CICS is to autoinstall undefined consoles automatically without any input from the autoinstall control program. The

4-character termid required for the console's TCT entry is generated by CICS, beginning with a ~ (logical not) symbol.

See the *CICS Customization Guide* for information about writing an autoinstall control program that supports consoles.

**AIXIT**={DFHZATDX | DFHZATDY | name}

specifies the name of the autoinstall user-replaceable program that you want CICS to use when autoinstalling local VTAM terminals, APPC connections, virtual terminals, and shipped terminals and connections. Autoinstall is the process of installing resource definitions automatically, using VTAM logon or BIND data, model definitions, and an autoinstall program.

**Note:** You can specify only one user-replaceable program on the AIXIT parameter. Which of the CICS-supplied programs (or customized versions thereof) that you choose depends on what combination of resources you need to autoinstall.

For background information about autoinstall, see in the *CICS Resource Definition Guide*.

### **DFHZATDX**

A CICS-supplied autoinstall user program. This is the default. It installs definitions for:

- Locally-attached VTAM terminals
- Virtual terminals used by the CICS Client products
- Remote shipped terminals
- Remote shipped connections

### **DFHZATDY**

A CICS-supplied autoinstall user program. It installs definitions for:

- Locally-attached VTAM terminals
- Local APPC connections
- Virtual terminals used by the CICS Client products
- Remote shipped terminals
- Remote shipped connections

**name** The name of your own customized autoinstall program, which may be based on one of the supplied sample programs. For programming information about writing your own autoinstall program, see the *CICS Customization Guide*.

**AILDELAY**={0 | hhmmss}

specifies the delay period that elapses after all sessions between CICS and an autoinstalled terminal, APPC device, or APPC system are ended, before the terminal or connection entry is deleted. All sessions are ended when the terminal or system logs off, or when a transaction disconnects it from CICS.

**Note:** The AILDELAY parameter does not apply to the following types of autoinstalled APPC connection, which are not deleted:

- Sync level 2-capable connections (for example, CICS-to-CICS connections)
- Sync level 1-only, limited resource connections installed on a CICS that is a member of a generic resource group

### **hhmmss**

A 1 to 6-digit number. The default is 0. For non-LU6.2 terminals and LU6.2 single-session connections installed by a CINIT, 0 means that the terminal entry is deleted as soon as the session is ended. For LU6.2

connections installed by a BIND, 0 means that the connection is deleted as soon as all sessions are ended, but is reusable if a new BIND occurs before the deletion starts.

If you leave out the leading zeros, they are supplied (for example, 123 becomes 000123—that is, 1 minute 23 seconds).

**AIQMAX={100|number}**

specifies the maximum number of VTAM terminals and APPC connections that can be queued concurrently for autoinstall.

**number**

A number in the range 0 through 999. The default is 100.

A zero value disables the autoinstall function.

Specify a number that is large enough to allow for both APPC connections and terminals.

**Note:** This value does not limit the total number of terminals that can be autoinstalled. If you have a large number of terminals autoinstalled, shutdown can fail due to the MXT system initialization parameter being reached or CICS becoming short on storage. For information about preventing this possible cause of shutdown failure, see the *CICS Customization Guide*.

**AIRDELAY={700|hmmss}**

specifies the delay period that elapses after an emergency restart before autoinstalled terminal and APPC connection entries that are not in session are deleted. The AIRDELAY parameter also applies when CEMT SET VTAM OPEN is issued after a VTAM abend and PSTYPE=MNPS is coded. This causes autoinstalled resources to be deleted, if the session was not restored and has not been used since the ACB was opened.

**Note:** The AIRDELAY parameter does not apply to the following types of autoinstalled APPC connection, which are always written to the CICS global catalog and recovered during a warm or emergency start:

- Sync level 2-capable connections (for example, CICS-to-CICS connections)
- Sync level 1-capable, limited resource connections installed on a CICS that is a member of a generic resource group

**hmmss**

A 1-to 6-digit number. If you leave out the leading zeros, they are supplied. The default is 700, meaning a delay of 7 minutes. A value of 0 means that autoinstalled definitions are not written to the global catalog and therefore are not restored at an emergency restart.

For guidance about the performance implications of setting different AIRDELAY values, see the *CICS Customization Guide*.

**Note:** If you are running CICS with XRF, set the same value on the AIRDELAY parameter for both the active and the alternate CICS regions. It is particularly important, if you want autoinstall sessions to be reestablished after a takeover, that you avoid coding a zero on this parameter for either the active or the alternate CICS regions.

For background information, see the *CICS/ESA 3.3 XRF Guide*.

**AKPFREQ={4000|number}**

specifies the number of write requests to the CICS system log stream output buffer required before CICS writes an activity keypoint. (For more information about activity keypointing, see the *CICS Customization Guide*.)

**4000**

This is the default.

*number*

*number* can be 0 (zero) or any value in the range 200 through 65535. (You cannot specify a number in the range 1—199.) You are recommended to allow AKPFREQ to assume its default value, 4000.

**Note:** If you specify AKPFREQ=0, no activity keypoints are written, with the following consequences:

- The CICS system log automatic deletion mechanism will not work so efficiently in this situation. The average system log occupancy would merely increase, maybe quite dramatically for some users. Without efficient automatic deletion, the log stream will spill onto secondary storage, and from there onto tertiary storage (unless you control the size of the log stream yourself).
- Emergency restarts are not prevented, but the absence of activity keypoints on the system log affects the performance of emergency restarts because CICS has to read backwards through the entire log stream.
- Backout-while-open (BWO) support is seriously affected, because without activity keypointing, tie-up records are not written to the forward recovery logs and the data set recovery point is not updated. Therefore, for forward recovery to take place, all forward recovery logs must be kept since the data set was first opened for update after the last image copy. For more information about the effect of AKPFREQ=0 on BWO, see “Effect of disabling activity keypointing” on page 33.

**APPLID={DBDCCICS|applid}**

specifies the VTAM application identifier for this CICS region.

**applid** This name, 1 through 8 characters, identifies the CICS region in the VTAM network. It must be unique if running in a sysplex. It must match the name field specified in the APPL statement of the VTAM VBUILD TYPE=APPL definition. For an example, see the *CICS Transaction Server for z/OS Installation Guide*.

When you define this CICS region to another CICS region, in a CONNECTION definition you specify the applid as the NETNAME. When sharing a DL/I database with a batch region, the applid is used by the batch region to identify the CICS region.

If the CICS region uses XRF, the form of the APPLID parameter is:

**APPLID=(generic\_applid,specific\_applid)**

specifies the generic and specific XRF applids for the CICS region. Both applids must be 1 through 8 characters.

**generic\_applid**

The generic applid for both (active and alternate) the active and the alternate CICS regions. Therefore, you must specify the same name for *generic\_applid* on the APPLID system initialization parameter for both CICS regions. Because IRC uses *generic\_applid* to identify the CICS regions, there can be no IRC connection for an alternate CICS region until takeover has occurred and the alternate CICS region becomes the active CICS region.

When you define this XRF pair to another CICS region, in a CONNECTION definition you specify the generic applid as the NETNAME.

When sharing a DL/I database with a batch region, this name is used by the batch region to identify the CICS region. CICS passes the generic applid to DBRC, because the alternate system does not sign on to DBRC until it has completed takeover.

Do not confuse the term *generic applid* with *generic resource name*. Generic applids apply only to CICS regions that use XRF. Generic resource names apply only to VTAM generic resource groups.

#### **specific\_applid**

specifies the CICS region in the VTAM network. It must match the label specified in the VTAM VBUILD TYPE=APPL definition. You must specify a different *specific\_applid* on the APPLID system initialization parameter for the active and for the alternate CICS region. Also, *generic\_applid* and *specific\_applid* must be different.

The active and alternate CICS regions use the VTAM MODIFY USERVAR command to set a user application name variable, so end users do not need to know which CICS region is active at any instant. For background information about using this command, see the *CICS/ESA 3.3 XRF Guide*.

#### **AUTCONN={0|hhmmss} (alternate)**

specifies that the reconnection of terminals after an XRF takeover is to be delayed, to allow time for manual switching. The delay is hh hours, mm minutes, ss seconds. The default value of zero means that there is no delay in the attempted reconnection.

The interval specified is the delay before the CXRE transaction runs. CXRE tries to reacquire any XRF-capable (class 1) terminal session that failed to get a backup session, or failed the switch for some other reason. CXRE tries to reacquire other terminals that were in session at the time of the takeover.

Note that the same delay interval applies to the connection of terminals with AUTOCONNECT(YES) specified in the TYPETERM definition, at a warm or emergency restart, whether or not you have coded XRF=YES.

#### **AUTODST={NO|YES}**

specifies whether CICS is to activate automatic dynamic storage tuning for application programs.

**NO** Automatic dynamic storage tuning is not required and CICS does not request this support from Language Environment.

**YES** Automatic dynamic storage tuning is required. This is activated during CICS startup when Language Environment is being initialized. CICS indicates to Language Environment that it is able to support dynamic storage tuning, and if Language Environment responds by indicating that it also supports the facility, CICS and Language Environment are synchronized to provide the required support.

For more information, see the appropriate OS/390 Language Environment manual.

#### **AUTORESETTIME={NO|YES}**

specifies the action CICS should take if, at the next local midnight, the CICS time-of-day differs from the system time-of-day by more than 30 minutes (for example, setting clocks forward or back to adjust for Summer and Winter time).

#  
#  
#  
#

- # **NO** CICS issues message DFHAP1500 to indicate that a CEMT PERFORM RESET is required to synchronize the CICS time-of-day with the system time-of-day.
- #
- #
- # **YES** CICS issues a PERFORM RESET to synchronize the CICS time-of-day with the system time-of-day
- #

**AUXTR={OFF|ON}**

specifies whether the auxiliary trace destination is to be activated at system initialization. This parameter controls whether any of the three types of CICS trace entry are written to the auxiliary trace data set. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (that are always made and are not controlled by a system initialization parameter).

**OFF** Do not activate auxiliary trace.

**ON** Activate auxiliary trace.

For details of internal tracing in main storage, see the INTTR parameter on page "INTTR" on page 205.

**AUXTRSW={NO|ALL|NEXT}**

specifies whether you want the auxiliary trace autoswitch facility.

**NO** Disables the autoswitch facility.

**NEXT** Enables the autoswitch facility to switch to the next data set at end of file of the first data set used for auxiliary trace. Coding NEXT permits one switch only, and when the second data set is full, auxiliary trace is switched off.

**ALL** Enable the autoswitch facility to switch to the inactive data set at every end of file. Coding ALL permits continuous switching between the two auxiliary trace data sets, DFHAUXT and DFHBUXT, and whenever a data set is full, it is closed and the other data set is opened.

**BMS=({MINIMUM|STANDARD|def.FULL }[,COLD][,{def.UNALIGN |ALIGN}] [, {DDS|NODDS}])**

specifies which version of basic mapping support you want to be included. The function included in each version of BMS is shown in Table 19 on page 170. The parameter BMS can be overridden during CICS initialization.

You need full or standard function BMS, if you are using XRF and have specified MESSAGE for RECOVNOTIFY on any of your TYPETERM definitions.

**MINIMUM**

The minimum version of BMS is included.

**STANDARD**

The standard version of BMS is included.

**FULL** The full version of BMS is included. This is the default in the SIT.

**COLD**

CICS deletes delayed messages from temporary storage, and destroys their interval control elements (ICEs). COLD forces the deletion of messages regardless of the value in effect for START. If COLD is not specified, the availability of messages will depend on the values in effect for the START and TS parameters.

**UNALIGN**

specifies that all BMS maps assembled before CICS/OS/VS Version 1

Release 6 are unaligned. Results are unpredictable if the stated alignment does not match the actual alignment.

**ALIGN**

All BMS maps assembled before CICS/OS/VS Version 1 Release 6 are aligned.

**DDS**

BMS is to load suffixed versions of map sets and partition sets. BMS first tries to load a version that has the alternate suffix (if the transaction uses the alternate screen size). If the load fails, BMS tries to load a version that has the default map suffix. If this fails too, BMS tries to load the unsuffixed version. DDS, which stands for “device dependent suffixing”, is the default.

You need to use map suffixes only if the same transaction is to be run on terminals with different characteristics (in particular, different screen sizes). If you do not use suffixed versions of map sets and partition sets, CICS need not test for them.

**NODDS**

BMS is not to load suffixed versions of map sets and partition sets. Specifying NODDS avoids the search for suffixed versions, saving processor time.

Table 19. Versions of BMS

BMS version	Devices supported	Function provided
MINIMUM	All 3270 system display units and printers except SNA character string printers, which are defined as DEVICE(SCSPRINT) on the RDO TYPETERM definition or as TRMTYPE=SCSPRT in DFHTCT	SEND MAP command, RECEIVE MAP command, SEND CONTROL command. Default and alternate screens; extended attributes; map set suffixes; screen coordination with null maps; and block data
STANDARD	All devices are supported by BMS. These are listed in the <i>CICS Application Programming Guide</i>	All function of MINIMUM, <b>plus</b> outboard formats, partitions, controlling a magnetic slot reader, NLEOM mode for 3270 system printers, SEND TEXT command, and Subsystem LDC controls.
FULL	All devices supported by BMS. These are listed in the <i>CICS Application Programming Guide</i>	Same as STANDARD, <b>plus</b> terminal operator paging, cumulative mapping, page overflow, cumulative text processing, routing, message switching returning BMS-generated data stream to program before output.

**BRMAXKEEPTIME={86400|number}**

This parameter specifies the maximum time (in seconds) that bridge facilities (virtual terminals used by the 3270 bridge) are kept if they are not used. The client application can specify this timeout value when it sends a request to run a transaction using the Link3270 bridge. If the client specifies a larger value than the BRMAXKEEPTIME value in the AOR, then CICS will change this parameter in the link parameter list.

**number**

The maximum timeout value that a client can specify (in seconds), before an unused bridge facility is deleted. The value specified must be in the range 0 to 86400. A value of 0 means that bridge facilities are never kept at the end of a transaction, therefore CICS will not be able to run pseudoconversational transactions. This may be useful if the region is only used for inquiry transactions. The default value is 24 hours (86400 seconds).

**CDSASZE={0K|number}**

specifies the size of the CDSA. The default size is 0, indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

specify number as an amount of storage in the range 0 to 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the CDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**CHKSTRM={CURRENT|NONE}**

specifies that terminal storage-violation checking is to be activated or deactivated.

**CURRENT**

TIOA storage violations are to be checked.

**NONE** TIOA storage-violation checking is to be deactivated.

You can also use the CICS-supplied transaction, CSFE, to switch terminal storage-violation checking on and off.

For information about checking for storage violations, see the *CICS Transaction Server for z/OS Installation Guide*.

**Restrictions** You can specify the CHKSTRM parameter in PARM, SYSIN, or CONSOLE only.

**CHKSTSK={CURRENT|NONE}**

specifies that task storage-violation checking at startup is to be activated or deactivated.

**CURRENT**

All storage areas on the transaction storage chain for the current task only are to be checked.

**NONE** Task storage-violation checking is to be deactivated.

You can also use the CICS-supplied transaction, CSFE, to switch task storage-violation checking on and off.

For information about checking for storage violations, see the *CICS Transaction Server for z/OS Installation Guide*.

**Restrictions** You can specify the CHKSTSK parameter in PARM, SYSIN, or CONSOLE only.

**CICSSVC={216|number}**

specifies the number that you have assigned to the CICS type 3 SVC. The default number is 216.

A CICS type 3 SVC with the specified (or default) number must be installed in the LPA. For information about installing the CICS SVC, see *CICS Transaction Server for z/OS Installation Guide*.

CICS checks if the SVC number supplied corresponds to the correct level of the CICS Type 3 SVC module, DFHCSVC. If the SVC number does not correspond to the correct level of DFHCSVC, the following can happen, depending on the value specified for the PARMERR system initialization parameter:

- CICS is terminated with a system dump
- The operator is allowed to retry using a different SVC number

For details of the PARMERR system initialization parameter, see page “PARMERR” on page 219.

**CILOCK={NO|YES}**

specifies whether or not the control interval lock of a non-RLS VSAM file is to be kept after a successful read-for-update request.

**NO** is the default and specifies that the control interval is to be freed. This allows other tasks to access other records in the same control interval, without an exclusive control conflict occurring. In these cases throughput should be greater. Note that the record lock on the record for which the read-for-update was first issued, still prevents other tasks from updating this record, even though the control interval lock has been released. When the record is rewritten or deleted, the read-for-update is reissued to VSAM as part of the update processing.

# If a WRITE is issued by another task during a READ UPDATE,  
# the WRITE receives a DUPREC condition.

**YES** specifies that the control interval is not to be freed. This means that a subsequent rewrite or delete request does not need to reissue the read-for-update request to VSAM. However, if other tasks attempt to access other records in the same control interval, an exclusive control conflict occurs on this control interval, forcing these tasks to wait until the update request completes.

**CLINTCP={437|codepage}**

specifies the default client code page to be used by the DFHCNV data conversion table but only if the CLINTCP parameter in the DFHCNV macro is set to SYSDEF. The *codepage* is a field of up to 8 characters and can take the values supported by the CLINTCP parameter in the DFHCNV macro. See the *CICS Family: Communicating from CICS on System/390* for the list of valid code pages. The default is 437.

**CLSDSTP={NOTIFY|NONOTIFY}**

specifies the notification required for an EXEC CICS ISSUE PASS command. This parameter is applicable to both autoinstalled and non-autoinstalled

terminals. You can use the notification in a user-written node error program to reestablish the CICS session when a VTAM CLSDST PASS request resulting from an EXEC CICS ISSUE PASS command fails. For more information about the EXEC CICS ISSUE PASS command, see the *CICS Application Programming Reference* manual.

**NOTIFY** CICS requests notification from VTAM when the EXEC CICS ISSUE PASS command is executed.

**NONOTIFY** CICS does not request notification from VTAM.

**CLT=xx (alternate)**

specifies the suffix for the command list table (CLT), if this SIT is used by an alternate XRF system. The name of the table is DFHCLTxx.

For information about coding the macros for this table, see the *CICS Application Programming Reference*.

**CMDPROT={YES|NO}**

specifies that you want to allow, or inhibit, CICS validation of start addresses of storage referenced as output parameters on EXEC CICS commands.

**YES** CICS validates the initial byte at the start of any storage that is referenced as an output parameter on EXEC CICS commands to ensure that the application program has write access to the storage. This ensures that CICS does not overwrite storage on behalf of the application program when the program itself cannot do so. If CICS detects that an application program has asked CICS to write into an area to which the application does not have addressability, CICS abends the task with an AEYD abend.

The level of protection against bad addresses depends on the level of storage protection in the CICS environment. The various levels of protection provided when you specify CMDPROT=YES are shown in Table 20.

**NO** CICS does not perform any validation of addresses of the storage referenced by EXEC CICS commands. This means that an application program could cause CICS to overwrite storage to which the application program itself does not have write access.

Table 20. Levels of protection provided by CICS validation of application-supplied addresses

Environment	Execution key of affected programs	Types of storage referenced by applications that cause AEYD abends
Read-only storage (RENTPGM=PROTECT)	CICS-key and user-key	CICS key 0 read-only storage (RDSA and ERDSA).
Subsystem storage protection (STGPROT=YES)	User-key	All CICS-key storage (CDSA and ECDSA)
Transaction isolation (TRANISO=YES)	User-key and ISOLATE(YES)	Task-lifetime storage of all other transactions
Transaction isolation (TRANISO=NO)	User-key and ISOLATE(NO)	Task-lifetime storage of all <b>except</b> other user key and ISOLATE(NO) transactions
Base CICS (all storage is CICS key 8 storage) (RENTPGM=NOPROTECT; STGPROT=NO; and TRANISO=NO)	CICS-key and user-key	MVS storage only

**CMDSEC={ASIS|ALWAYS}**

specifies whether or not you want CICS to honor the CMDSEC option specified on a transaction's resource definition.

**ASIS** means that CICS honors the CMDSEC option defined in a transaction's resource definition. CICS calls its command security checking routine only when CMDSEC=YES is specified in a transaction resource definition.

**ALWAYS** CICS overrides the CMDSEC option, and always calls its command security checking routine to issue the appropriate call to the SAF interface.

**Notes:**

1. Specify ALWAYS when you want to control the use of the SPI in all your transactions. Be aware that this might incur additional overhead. The additional overhead is caused by CICS issuing the command security calls on every eligible EXEC CICS command, which are *all* the system programming interface (SPI) commands.
2. If you specify ALWAYS, command checking applies to CICS-supplied transactions such as CESN and CESH. You must authorize all users of CICS-supplied transactions to use the internal CICS resources for the transactions, otherwise you will get unexpected results in CICS-supplied transactions.

**Restrictions** You can specify the CMDSEC parameter in the SIT, PARM, or SYSIN only.

**CONFDATA={SHOW|HIDETC}**

specifies whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps that contain the VTAM receive any input area (RAIA). This option applies to initial input data received on a VTAM RECEIVE ANY operation, the initial input data received on an MRO link, and FEPI screens and RPLAREAs.

This option also applies to the CICS client use of a Virtual Terminal. Data is traced before and after codepage conversion and is suppressed if HIDETC is used in combination with CONFDATA YES in the transaction.

**SHOW** Data suppression is not in effect. User data is traced regardless of the CONFDATA option specified in transaction resource definitions. This option overrides the CONFDATA option in transaction resource definitions.

**HIDETC** CICS is to 'hide' user data from CICS trace entries. It also indicates that VTAM RAIAs are to be suppressed from CICS dumps. The action actually taken by CICS is subject to the individual CONFDATA attribute on the transaction resource definition (see Table 21 on page 176).

If you specify CONFDATA=HIDETC, CICS processes VTAM, MRO, and FEPI user data as follows:

- **VTAM:** CICS clears the VTAM RAIA containing initial input as soon as it has been processed, and before the target transaction has been identified.

The normal trace entries (FC90 and FC91) are created on completion of the RECEIVE ANY operation with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data except the first 4 bytes of normal data, or the first 8 bytes of function management headers (FMHs).

CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from the FC90 trace in the trace entry AP FC92. This trace entry is not created if the transaction is defined with CONFDATA(YES).

- **MRO:** CICS does not trace the initial input received on an MRO link.

The normal trace entries (DD16, DD23, and DD25) are created with the text "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT" replacing all the user data.

CICS then identifies the target transaction for the data. If the transaction definition specifies CONFDATA(NO), CICS traces the user data that it suppressed from DD16 in the trace entry AP FC92. This special trace entry is not created if the transaction is defined with CONFDATA(YES).

- **FEPI:** FEPI screens and RPL data areas (RPLAREAs) areas are suppressed from all FEPI trace points if CONFDATA(YES) is specified in the transaction resource definition. The user data in the FEPI trace points AP 1243, AP 1244, AP 145E, AP 145F, AP 1460, AP 1461, AP 1595, AP 1596, AP 1597, AP 1598, and AP 1599 is replaced with the message "SUPPRESSED DUE TO CONFDATA=HIDETC IN SIT". If the transaction definition specifies CONFDATA(NO), the FEPI trace entries are created with the user data as normal.

**Mirror transactions:** The CICS-supplied mirror transaction definitions are specified with CONFDATA(YES). This ensures that, when you specify CONFDATA=HIDETC as a system initialization parameter, CICS regions running mirror transactions suppress user data as described for VTAM and MRO data.

**Modified data:** By waiting until the transaction has been identified to determine the CONFDATA option, VTAM or MRO data may have been modified (for example, it may have been translated to upper case).

The interaction between the CONFDATA system initialization parameter and the CONFDATA attribute on the transaction resource definition is shown in Table 21 on page 176.

Table 21. Effect of CONFDATA system initialization and transaction definition parameters

CONFDATA on transaction	CONFDATA system initialization parameter	
	SHOW	HIDETC
NO	Data not suppressed	VTAM RAIAs are cleared. Initial input of VTAM and MRO data is suppressed from the normal FC90, FC91, DD16, DD23, and DD25 trace entries. For FC90 and DD16 traces only, suppressed user data is traced separately in an FC92 trace entry. FEPI screens and RPLAREAs are traced as normal.
YES	Data not suppressed	VTAM RAIAs are cleared. All VTAM, MRO, and FEPI user data is suppressed from trace entries.

You cannot modify the CONFDATA option while CICS is running. You must restart CICS to make such a change.

**Restrictions** You can specify the CONFDATA parameter in the SIT, PARM, and SYSIN only.

**CONFTXT={NO|YES}**

specifies whether CICS is to prevent VTAM from tracing user data.

**NO** CICS does not prevent VTAM from tracing user data.

**YES** CICS prevents VTAM from tracing user data.

**Restrictions** You can specify the CONFTEXT parameter in the SIT, PARM, and SYSIN only.

**CPSMCONN={NO|CMAS|LMAS|WUI}**

specifies whether you want CICS to invoke the specified CICSplex<sup>®</sup> SM component to initialize the region as one of the following:

- A CICSplex SM address space (CMAS)
- A CICSplex SM managed application system (MAS)
- A CICSplex SM Web User Interface server

**NO** Do not invoke any CICSplex SM initialization code in this region.

**CMAS** Invoke CICSplex SM code automatically during CICS initialization to initialize the region as a CMAS. The other information CICSplex SM needs for a CMAS is taken from the CMAS parameters read from the EYUPARM data set, and from resource definitions installed from the CSD from group list EYU310L0.

Specifying CPSMCONN=CMAS is the recommended alternative to specifying the CICSplex SM CMAS initialization program in a CICS post-initialization program list table (PLTPI).

**Note:** If you specify CPSMCONN=CMAS, ensure that your CICS region startup JCL EXEC statement specifies the name of the CICSplex SM CMAS program, EYU9XECS. For example:

```
//CMAS EXEC PGM=EYU9XECS,.....
```

**LMAS** Invoke CICSplex SM code automatically during CICS

initialization to initialize the region as a local MAS. The other information CICSplex SM needs for a MAS is taken from the MAS parameters read from the EYUPARM data set, and from resource definitions installed from the CSD from group EYU310G1.

Specifying CPSMCONN=LMAS is the recommended alternative to specifying the CICSplex SM MAS initialization program in a CICS post-initialization program list table (PLTPI).

## WUI

Invoke CICSplex SM code automatically during CICS initialization to initialize the region as a CICSplex SM Web User Interface server. The other information CICSplex SM needs is taken from the MAS and WUI parameters read from the EYUPARM and EYUWUI data sets respectively, and from resource definitions installed from the CSD from groups EYU310G1 and EYU310GW.

Specifying CPSMCONN=WUI is the recommended alternative to specifying the CICSplex SM MAS and WUI initialization and shutdown programs in initialization and shutdown program list tables (PLTPI and PLTSD).

Note that using the CPSMCONN parameter has the same effect as specifying the relevant CICSplex SM program in a program list table. This means that MASPLTWAIT and other PLT-related CICSplex SM parameters are still valid and should be specified as necessary.

For information about starting CICSplex SM address spaces, see the *CICS Transaction Server for z/OS Installation Guide*.

#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#

## CRLPROFILE=PROFILENAME

Specifies the 246-character uppercase name of the profile that will be used to authorize CICS to access the certification revocation lists (CRLs) that are stored in an LDAP server. The profile name is specified in the external security manager's LDAPBIND general resource class that contains bind information for an LDAP server.

The profile must contain the name of the LDAP server and the distinguished name and password of a user who is authorized to extract certification revocation lists from it. For more information about setting up the profile, see the *CICS RACF Security Guide*. Specifying this parameter means that CICS checks each client certificate during the SSL negotiation for a revoked status using the certificate revocation lists in the LDAP server. If the certificate is revoked, CICS closes the connection immediately.

If the CRLPROFILE parameter is omitted or invalid, or the specified profile contains invalid data, or if the LDAP server identified by the profile is unavailable, CICS does not check the revoked status of certificates during SSL handshakes.

## CSDACC={READWRITE|READONLY}

specifies the type of access to the CSD to be permitted to this CICS region. Note that this parameter is effective only when you start CICS with a START=COLD parameter. If you code START=AUTO, and CICS performs a warm or emergency restart, the file resource definitions for the CSD are recovered from the CICS global catalog. However, you can redefine the type of access permitted to the CSD dynamically with a CEMT SET FILE, or an EXEC CICS SET FILE, command.

### **READWRITE**

Read/write access is allowed, permitting the full range of CEDA, CEDB, and CEDC functions to be used.

### **READONLY**

Read access only is allowed, limiting the CEDA and CEDB transactions to only those functions that do not require write access.

### **CSDBKUP={STATIC|DYNAMIC}**

specifies whether or not the CSD is eligible for BWO. If BWO is wanted, specify CSDBKUP=DYNAMIC.

The CSDBKUP, CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see “Planning for backup and recovery” on page 76.

### **STATIC**

All CICS files open for update against the CSD data set must be quiesced before a DFHSM and DFDSS backup of the CSD data set. The files must remain quiesced during the backup.

### **DYNAMIC**

DFHSM and DFDSS are allowed to make a data set back up copy while CICS is updating the CSD.

Note that CSDBKUP=DYNAMIC is valid only if you have also specified CSDRECOV=ALL.

### **CSDBUFND=number**

specifies the number of buffers to be used for CSD data. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter plus 1, up to a maximum of 32768. Note that this parameter is used only if you have also coded CSDLRNO=NONE; if you have coded CSDLRNO=number, CSDBUFND is ignored.

If you specify a value for CSDBUFND that is less than the required minimum (the CSDSTRNO value plus 1), VSAM automatically changes the number of buffers to the number of strings plus 1 when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

### **CSDBUFNI=number**

specifies the number of buffers to be used for the CSD index. The minimum you should specify is the number of strings coded on the CSDSTRNO parameter, up to a maximum of 32768. This parameter is used only if you have also coded CSDLRNO=NONE; if you have coded CSDLRNO=number, CSDBUFNI is ignored.

If you specify a value for CSDBUFNI that is less than the required minimum (the CSDSTRNO value), VSAM automatically changes the number of buffers to the number of strings when CICS issues the OPEN macro for the CSD.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

### **CSDDISP={OLD|SHR}**

specifies the disposition of the data set to be allocated to the CSD. If no JCL

statement for the CSD exists when it is opened, the open is preceded by a dynamic allocation of the CSD using this disposition. If a DD statement exists in the JCL of the CICS startup job, it takes precedence over this disposition.

**OLD** The disposition of the CSD is set to OLD if dynamic allocation is performed.

**SHR** The disposition of the CSD is set to SHR if dynamic allocation is performed.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

**CSDDSN=name**

specifies the 1- to 44-character JCL data set name (DSNAME) to be used for the CSD. If no JCL statement exists for the CSD when it is opened, the open is preceded by a dynamic allocation of the CSD using this DSNAME. If a DD statement exists in the JCL of the CICS startup job, it takes precedence over this DSNAME.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

**CSDFRLOG=number**

specifies a number that corresponds to the journal name that CICS uses to identify the forward recovery log stream for the CSD.

This parameter is meaningful only if CSDRECOV=ALL and CSDRLS=NO are specified, otherwise it is ignored. If you specify CSDRLS=NO and CSDRECOV=ALL, but omit CSDFRLOG (or specify CSDFRLOG=NO), the SIT assembly fails. However, if you specify an invalid combination as SIT overrides, CICS initialization will fail.

CSDBKUP, CSDRECOV and CSDFRLOG are ignored if CSDRLS=YES is specified. This is because recovery attributes (that is, the recoverability, the forward recovery LSN, and the BWO eligibility) must be specified in the ICF catalog for data sets that are opened in RLS mode.

The recovery attributes can also be specified (optionally) in the ICF catalog when you specify CSDRLS=NO. If you specify recovery attributes in both the ICF catalog and as system initialization parameters, the ICF catalog values are used (but see the next paragraph).

For a CSD opened in a non-RLS mode (CSDRLS=NO), the CSDBKUP, CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified. For information about their effects when the SIT is assembled and during CICS override processing, see “Planning for backup and recovery” on page 76.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

**number**

The journal number that identifies the user journal that CICS is to use for forward recovery of the CSD. CICS journal names are of the form DFHJ $nn$  where  $nn$  is a number in the range 1 through 99. CICS maps the resulting journal name (DFHJ01—DFHJ99) to an MVS log stream.

**CSDINTEG={UNCOMMITTED|CONSISTENT|REPEATABLE}**

specifies the level of read integrity for the CSD if it is accessed in RLS mode. If the CSD is not accessed in RLS mode (CSDRLS=NO), a value for CSDINTEG of CONSISTENT or REPEATABLE will be changed to UNCOMMITTED.

**UNCOMMITTED**

The CSD is read without read integrity. For each read request, CICS obtains the current value of the record as known to VSAM. No attempt is made to serialize this read request with any concurrent update activity for the same record. The record returned may be a version updated by another RDO task but not yet committed, and this record could change if the update is subsequently backed out.

**CONSISTENT**

CICS reads the CSD with consistent read integrity. If a record is being modified by another RDO task, the READ request waits until the update is complete, the timing of which depends on whether the CSD is recoverable or non-recoverable:

- For a recoverable CSD, the READ request completes when the updating transaction completes its next syncpoint or rollback.
- For a non-recoverable CSD, the READ completes as soon as the VSAM request performing the update completes.

**REPEATABLE**

CICS reads the CSD with repeatable read integrity. If the record is being modified by another RDO task, the READ request waits until the update is complete, the timing of which depends on whether the CSD is recoverable or non-recoverable:

- For a recoverable CSD, the READ request completes when the updating transaction completes its next syncpoint or rollback.
- For a non-recoverable CSD, the READ completes as soon as the VSAM request performing the update completes.

After the CSD read completes, a shared lock remains held until syncpoint. This guarantees that a CSD record read within an RDO task cannot be modified until the end of the task (for example, a CEDA transaction) that is reading the CSD.

**CSDJID={NO|number}**

specifies the journal identifier of the journal that you want CICS to use for automatic journaling of file requests against the CSD.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

**NO** You do **not** want automatic journaling for the CSD. This is the default.

**number**

A number in the range 1 through 99 to identify the journal that CICS is to use for automatic journaling for the CSD. Mapping to a log stream works in the same way that CSDFRLOG does, that is, *nn* maps to DFHJnn. 01 no longer maps to the system log.

The automatic journaling options enforced for the CSD when you code CSDJID=number are JNLADD=BEFORE and JNLUPDATE=YES. These options are sufficient to record enough information for a user-written forward recovery utility. No other automatic journaling options are

available for the CSD. For information about the options JNLADD=BEFORE and JNLUPDATE=YES, see the *CICS Resource Definition Guide*.

**CSDLSRNO**={1 | number | NONE | NO}

specifies whether the CSD is to be associated with a local shared resource (LSR) pool.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the LSR pool attribute for the CSD dynamically with an EXEC CICS SET FILE command.

1 The default LSR pool number is 1.

**number**

The number of the LSR pool the CSD is to be associated with. The number of the pool must be in the range 1 through 8.

**NONE|NO**

The CSD is not to be associated with a local shared resource pool.

**CSDRECOV**={NONE | ALL | BACKOUTONLY}

specifies whether the CSD is a recoverable file.

The CSDBKUP, CSDRECOV, and CSDFRLOG system initialization parameters interact according to how they are specified, if CSDRLS=NO is specified. If CSDRLS=YES is specified, these parameters are ignored, because the recovery attributes must be specified in the VSAM catalog (using the BWO, LOG, and LOGSTREAMID parameters on DEFINE CLUSTER or ALTER CLUSTER). If CSDRLS=NO is specified but LOG has been specified in the VSAM catalog, the recovery attributes are taken from the VSAM catalog, and CSDBKUP, CSDRECOV, and CSDFRLOG do not need to be specified. If they are specified, however, the rules given in “Planning for backup and recovery” on page 76 must still be followed.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog.

**NONE** The CSD is not recoverable.

**ALL** You want both forward recovery and backout for the CSD. If you code ALL, also specify CSDFRLOG with the journal identification of the journal to be used for forward recovery of the CSD.

**Note:** If the journal you specify for logstreams associated with CSD recovery (CSDJID, CSDFRLOG, and possibly the log of logs, DFGLGLOG) is a DASD-only log stream, there can be delays when you use the CEDA transaction if the log stream requires a new connection. This delay is because the MVS system logger is formatting the staging dataset. Symptoms of the problem are:

```
DFHLG0771 07/08/01 03:30:42 IY0T1 A temporary error condition occurred
during MVS logger operation IXGWRITE for logstream xxxxxx.yyyyyy.zzzzzz.
MVS logger codes: X'00000008', X'00000868'.
```

If the CSD is the only file using those logstreams, CICS disconnects from the log when you end the CEDA transaction. The next time you run a CEDA transaction, CICS reconnects to the log stream and the MVS system logger allocates and formats a new staging data set.

## BACKOUTONLY

CSD recovery is limited to file backout only. If you specify backout for the CSD, CICS uses the system log to record before images for backout purposes.

## CSDRLS

specifies whether CICS is to access the CSD in RLS mode.

**NO** The CSD is opened in non-RLS mode, as specified on the CSDLSRNO parameter.

**YES** The CSD is opened in RLS mode. This enables you to update the CSD concurrently from several CICS regions, provided all the regions specify CSDRLS=YES. If a CICS region opens the CSD in RLS mode, another CICS region cannot open it in non-RLS mode. The first CICS region to open the CSD in a sysplex with SMSVSAM determines the access mode for all regions.

Your CSD must be defined to support RLS access: the IMBED option must not be specified, and recovery attributes must be defined in the VSAM catalog. The *CICS Transaction Server for z/OS Installation Guide* explains the data set characteristics required to support RLS access. If your CSD does not meet these requirements, it will fail to open.

If you specify both RLS and local shared resource (CSDLSRNO=*number*), RLS takes precedence.

If you specify CSDRLS=YES, the CSDRECOV, CSDFRLOG, and CSDJID parameters are ignored. You must specify the recovery attributes for an RLS-mode CSD in the ICF catalog entry for the CSD.

**Note:** If you define a recoverable CSD for RLS-mode access, you have to quiesce all RLS activity against the CSD before you can update the CSD using the batch utility program, DFHCSDUP. You can use the SET DSNAME QUIESCE command to do this, to ensure that no CEDA, CEDB, or CEDC transactions can run until you unquiesce the data set on completion of the batch job.

## CSDSTRNO={6|*number*}

specifies the number of concurrent requests that can be processed against the CSD. When the number of requests reaches the STRNO value, CICS automatically queues any additional requests until one of the active requests terminates.

CICS requires two strings per CSD user, and you can increase the CSDSTRNO value, in multiples of two, to allow more than one concurrent CEDA user.

See “Multiple users of the CSD within a CICS region (non-RLS)” on page 69 before you code this parameter.

This parameter is effective only on a CICS cold or initial start. On a warm or emergency restart, file resource definitions for the CSD are recovered from the global catalog. However, you can redefine the number of strings for the CSD dynamically with an EXEC CICS SET FILE command.

**6** The minimum number of concurrent requests for the CSD is 6.

### **number**

This number must be a multiple of 2, in the range 6 through 254.

## CWAKEY={USER|*CICS*}

specifies the storage key for the common work area (CWA) if you are operating

CICS with storage protection (STGPROT=YES). (You specify how much storage you want for the CWA on the WRKAREA parameter.) The permitted values are USER (the default), or CICS:

**USER** CICS obtains storage for the CWA in user key. This allows a user program executing in any key to modify the CWA.

**CICS** CICS obtains storage for the CWA in CICS key. This means that only programs executing in CICS key can modify the CWA, and user-key programs have read-only access.

If CICS is running without storage protection, the CWAKEY parameter is ignored, and the CWA is always allocated from CICS-key storage.

**DAE={NO|YES}**

specifies the default DAE action when new system dump table entries are created.

**NO** New system dump table entries will be created with DAEOPTION(NODAE). This means that the system dump will not be suppressed by the MVS Dump Analysis and Elimination (DAE) component.

**YES** New system dump table entries will be created with DAEOPTION(DAE). This means that the system dump is eligible for suppression by the MVS DAE component.

For more information about the DAEOPTION option, see *CICS System Programming Reference*.

**DATFORM={MMDDYY|DDMMYY|YYMMDD}**

specifies the external date display standard that you want to use for CICS date displays. An appropriate indicator setting is made in the CSA. It is examined by CICS supplied system service programs that display a Gregorian date. CICS maintains the date in the form 0CYYDDD in the CSA (where C=0 for years 19xx, 1 for years 20xx, and so on; YY=year of century; and DDD=day of year), and converts it to the standard you specify for display.

The DATFORM option selects the order in which the date is to be displayed. It does not select the format of the year. Both YY and YYYY formats are displayed.

**MMDDYY**

The date is in the form of month-day-year, MMDDYY and MMDDYYYY.

**DDMMYY**

The date is in the form of day-month-year, DDMMYY and DDMMYYYY.

**YYMMDD**

The date is in the form of year-month-day, YYMMDD and YYYYMMDD.

**DB2CONN={NO|YES}**

specifies whether you want CICS to start the DB2 connection automatically during initialization.

**NO** Do not automatically invoke DFHD2CM0, the CICS DB2 attach program, during initialization.

**YES** Invoke the CICS DB2 attach program, DFHD2CM0, automatically during CICS initialization. The other information CICS needs for starting the attachment is taken from CICS DB2 connection resource definitions installed from the CSD.

Specifying DB2CONN=YES is the recommended alternative to specifying the CICS DB2 attach programs in the CICS post-initialization program list table (PLT).

**DBCTLCON={NO|YES}**

specifies whether you want CICS to start the DBCTL connection automatically during initialization.

**NO** Do not automatically invoke DFHDBCON, the CICS DBCTL attach program, during initialization.

**YES** Invoke the CICS DBCTL attach program, DFHDBCON, automatically during CICS initialization. The other information CICS needs for starting the attachment, such as the DRA startup table suffix or the DBCTL subsystem name, is taken from an INITPARM system initialization parameter.

Specifying DBCTLCON=YES means you don't need to define the DBCTL attach program in the CICS post-initialization program list table (PLT).

**DEBUGTOOL={NO|YES}**

Specifies whether you want to use debugging profiles to select the programs that will run under the control of a debugging tool. The following debugging tools use debugging profiles:

- Debug Tool, for compiled language application programs (programs written in COBOL, PL/I, C, C++ and Assembler)
- Remote debugging tools (for compiled language application programs and Java programs)

Other debugging mechanisms, such as the CICS Execution Diagnostic Facility (CEDF) do not use debugging profiles.

**NO** Specifies that you do not want to use CICS debugging profiles to select the programs that will run under the control of a debugger tool.

**YES**

Specifies that you want to use CICS debugging profiles to select the programs that will run under the control of a debugging tool.

For more information, see the *CICS Application Programming Guide*.

**DFLTUSER={CICSUSER|userid}**

specifies the RACF userid of the default user; that is, the user whose security attributes are used to protect CICS resources in the absence of other, more specific, user identification. For example, except in the case of terminals defined with preset security, the security attributes of the default user are assigned to terminal users who do not sign on.

The specified userid must be defined to RACF if you are using external security (that is, you have specified the system initialization parameter SEC=YES).

The specified userid is signed on during CICS initialization. If it cannot be signed on, CICS fails to initialize.

**Restrictions** You can specify the DFLTUSER parameter in the SIT, PARM, or SYSIN only.

**DIP={NO|YES}**

specifies whether the batch data interchange program, DFHDIP, is to be included. This supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. (Support is provided for the transmit, print, message, user, and dump data sets

of the 3790 system.) (For the effect of this parameter, see page “Defining CICS resource table and module keywords” on page 162.)

**DISMACP={YES|NO}**

specifies whether CICS is to disable any transaction that terminates abnormally with an ASRD or ASRE abend (caused by a user program invoking a CICS macro, or referencing the CSA, the TCA, or the DB2 RCT).

**Note:** DISMACP=YES has no effect if the ASRD or ASRE abend is handled by an active abend exit.

**DOCCODEPAGE={037|codepage}**

specifies the default host code page to be used by the document domain. The *codepage* is a field of up to 8 characters. If *codepage* value is not specified, the default *doccodepage* is set to 037. See the *CICS Family: Communicating from CICS on System/390* for the list of valid code pages.

**DSALIM={5M|number}**

specifies the upper limit of the total amount of storage within which CICS can allocate the individual dynamic storage areas (DSAs) that reside below the 16MB boundary.

**5M** The default DSA limit is 5MB (5 242 880).

**number**

This is the amount of storage in the range 2MB to 16MB (2 097 152 bytes to 16 777 216 bytes) in multiples of 262 144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4 194 304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

From the storage size that you specify on the DSALIM parameter, CICS allocates the following dynamic storage areas:

**The user DSA (UDSA)**

The user-key storage area for all user-key task-lifetime storage below the 16MB boundary.

**The read-only DSA (RDSA)**

The key-0 storage area for all reentrant programs and tables below the 16MB boundary.

**The shared DSA (SDSA)**

The user-key storage area for any non-reentrant user-key RMODE(24) programs, and also for any storage obtained by programs issuing EXEC CICS GETMAIN commands for storage below the 16MB boundary with the SHARED option.

**The CICS DSA (CDSA)**

The CICS-key storage area for all non-reentrant CICS-key RMODE(24) programs, all CICS-key task-lifetime storage below the 16MB boundary, and for CICS control blocks that reside below the 16MB boundary.

**Notes:**

1. CICS allocates the UDSA in multiples of 1MB when transaction isolation is active, but in multiples of 256KB in CICS regions without transaction isolation. The other DSAs below 16MB are allocated in multiples of 256KB, with or without transaction isolation. The maximum you can specify depends on a number of factors, such as how you have configured your MVS

storage (which governs how much private storage remains below the line) and how much private storage you must leave free to satisfy MVS GETMAIN requests for storage outside the DSAs.

2. For information about calculating the amount of storage to specify on the DSALIM parameter, see the *CICS Performance Guide*.
3. Dynamic changes to the DSA limit are cataloged in the *local* catalog, and override the DSALIM parameter, if it is specified in the system initialization table, during all forms of restart-initial, cold, and warm. The cataloged value is *not* used if:
  - You specify startup values as system initialization parameters overrides (for example, in SYSIN).
  - You re-initialize the CICS catalog data sets.

**DSHIPIDL={020000|hmmss}**

specifies the minimum time, in hours, minutes, and seconds, that an *inactive* shipped terminal definition must remain installed in this region. When the timeout delete mechanism is invoked, only those shipped definitions that have been inactive for longer than the specified time are deleted.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to prevent terminal definitions having to be reshipped because they have been deleted prematurely.

The default minimum idle time is 2 hours.

#### **hmmss**

A 1- to 6-digit number in the range 0-995959. Numbers that have fewer than six digits are padded with leading zeros.

**DSHIPINT={120000|0|hmmss}**

specifies the interval between invocations of the timeout delete mechanism. The timeout delete mechanism removes any shipped terminal definitions that have not been used for longer than the time specified by the DSHIPIDL parameter.

You can use this parameter in a transaction routing environment, on the application-owning and intermediate regions, to control:

- How often the timeout delete mechanism is invoked
- The approximate time of day at which a mass delete operation is to take place, relative to CICS startup

**Note:** For more flexible control over when mass delete operations take place, you can use a CEMT SET DELETSHIPED or EXEC CICS SET DELETSHIPED command to reset the interval. (The revised interval starts *from the time the command is issued*, **not** from the time the remote delete mechanism was last invoked, nor from CICS startup.)

- 0** The timeout delete mechanism is not invoked. You might set this value in a terminal-owning region, or if you are not using shipped definitions.

#### **hmmss**

A 1- to 6-digit number in the range 1-995959. Numbers that have fewer than six digits are padded with leading zeros.

**DSRTPGM={NONE|DFHDSRP|program-name|EYU9XLOP}**

specifies the name of the distributed routing program to be used for dynamically routing:

- Eligible CICS business transaction services (BTS) processes and activities

For information about which BTS processes and activities are eligible for dynamic routing, see the *CICS Business Transaction Services*.

- Eligible non-terminal-related EXEC CICS START requests.

For information about which non-terminal-related START requests are eligible for dynamic routing, see the *CICS Intercommunication Guide*.

#### **DFHDSRP**

The CICS sample distributed routing program.

#### **EYU9XLOP**

The CICSplex SM routing program.

**NONE** For eligible CICS BTS processes and activities, no routing program is invoked. BTS processes and activities cannot be dynamically routed.

For eligible non-terminal-related START requests, the CICS sample distributed routing program, DFHDSRP, is invoked.

#### **program-name**

The name of a user-written program.

**Note:** See also the DTRPGM parameter, used to name the dynamic routing program.

#### **DTRPGM={DFHDYP|program-name}**

specifies the name of the dynamic routing program to be used for dynamically routing:

- Transactions initiated from user terminals
- Transactions initiated by eligible terminal-related EXEC CICS START commands
- Eligible program-link requests.

DFHDYP, the default, is the name of the CICS-supplied program. For information about which transactions started by EXEC CICS START commands, and which program-link requests, are eligible for dynamic routing, see the *CICS Intercommunication Guide*.

**Note:** See also the DSRTPGM parameter, used to name the distributed routing program.

#### **DTRTRAN={CRTX|name|NO}**

specifies the name of the transaction definition that you want CICS to use for dynamic transaction routing. This is intended primarily for use in a CICS terminal-owning region, although you can also use it in an application-owning region when you want to daisy-chain transaction routing requests. In a dynamic transaction routing environment, the transaction named on DTRTRAN must be installed in the CICS terminal-owning regions if you want to eliminate the need for resource definitions for individual transactions.

**Note:** DTRTRAN does not apply to non-terminal EXEC CICS START requests where the distributed routing program is invoked.

The transaction name is stored in the catalog for recovery during CICS restarts.

**CRTX** This is the default dynamic transaction definition. It is the name of the CICS-supplied sample transaction resource definition provided in the CSD group DFHISC.

**name** The name of your own dynamic transaction resource definition that you want CICS to use for dynamic transaction routing.

**NO** The dynamic transaction routing program is not invoked when a transaction definition cannot be found.

For information about the CICS-supplied sample transaction resource definition, CRTX, and about defining your own dynamic transaction routing definition, see Links and sessions in the *CICS Resource Definition Guide*.

**DUMP={YES|NO} (active and alternate)**

specifies whether the CICS dump domain is to take SDUMPs.

**YES** SDUMPs are produced, unless suppressed by the options specified in the CICS system dump table or by the MVS system defaults.

**NO** SDUMPs are suppressed.

**Note:** This does not prevent the CICS kernel from taking SDUMPs.

For more information about SDUMPs, see “System dumps” on page 99.

**DUMPDS={AUTO|A|B}**

specifies the transaction dump data set that is to be opened during CICS initialization.

**AUTO**

For all emergency or warm starts, CICS opens the transaction dump data set that was **not** in use when the previous CICS run terminated. This information is obtained from the CICS local catalog.

If you specify AUTO, or let it default, code DD statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

**A** CICS opens transaction dump data set DFHDMPA.

**B** CICS opens transaction dump data set DFHDMPB.

**DUMPSW={NO|NEXT}**

specifies whether you want CICS to switch automatically to the next dump data set when the first is full.

**NO** Disables the CICS autoswitch facility. If the transaction dump data set opened during initialization becomes full, CICS issues a console message to notify the operator. If you want to switch to the alternate data set, you must do so manually using the CEMT or EXEC CICS SET DUMPDS SWITCH command.

**NEXT** Enables the autoswitch facility to switch to the next data set at end of file of the data set opened during initialization. Coding NEXT permits one switch only. If you want to switch to the alternate data set again, you must do so manually using CEMT or EXEC CICS SET DUMPDS SWITCH command. If you specify NEXT, code DD statements for both of the transaction dump data sets, DFHDMPA and DFHDMPB, in your CICS startup job stream.

For more information about transaction dump data sets, see page 101.

**DURETRY={30|number-of-seconds|0}**

specifies, in seconds, the total time that CICS is to continue trying to obtain a system dump using the SDUMP macro. DURETRY allows you to control whether, and for how long, CICS is to reissue the SDUMP macro if another address space in the same MVS system is already taking an SDUMP when CICS issues an SDUMP request.

In the event of an SDUMP failure, CICS responds, depending on the reason for the failure, as follows:

- If MVS is already taking an SDUMP for another address space, and the DURETRY parameter is nonzero, CICS issues an MVS STIMERM macro to wait for five seconds, before retrying the SDUMP. CICS issues a message to say that it is waiting for five seconds before retrying the SDUMP. After five seconds CICS issues another message to say that it is retrying the SDUMP request.
- If the SDUMP fails for any other reason, such as no SYS1.DUMP data sets being available, or I/O errors preventing completion of the dump, CICS issues a message to inform you that the SDUMP has failed, and to give the reason why.

**30** 30 seconds allows CICS to retry up to 6 times (once every 5 seconds), if the cause of failure is that another region is taking an SDUMP.

**number-of-seconds**

Code the total number of seconds (up to 32767) during which you want CICS to continue retrying the SDUMP macro if the reason for failure is that another region is taking an SDUMP. CICS retries the SDUMP, once every five seconds, until successful or until retries have been made over a period equal to or greater than the DURETRY value.

**0** Code a zero value if you do not want CICS to retry the SDUMP macro.

**ECDSASZE={0K|number}**

specifies the size of the ECDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

Specify *number* as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

#  
#

**Note:** For the DS domain function CHANGE\_MODE, a trace entry will be generated if DS level 2, 3, or ALL tracing is active.

**Restrictions** You can specify the ECDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**EDSALIM={30M|number}**

specifies the upper limit of the total amount of storage within which CICS can allocate the individual extended dynamic storage areas (EDSAs) that reside above the 16MB boundary.

**30M** The default EDSA limit is 30MB (31 457 280 bytes).

**number**

Specify *number* as a value in the range 10MB to 2047MB, in multiples of 1MB. If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify *number* in bytes (for example, 33 554 432), or as a whole number of kilobytes (for example, 32 768K), or a whole number of megabytes (for example, 32M).

The maximum value allowed depends on a number of factors, such as:

- The size of the region you have specified on the MVS REGION parameter in the CICS job or procedure
- How much storage you require for the CICS internal trace table
- How much private storage you must leave free to satisfy MVS GETMAIN requests for storage above the 16MB boundary outside the DSAs

From the storage value that you specify on the EDSALIM parameter, CICS allocates the following extended dynamic storage areas:

**The extended user DSA (EUDSA)**

The user-key storage area for all user-key task-lifetime storage above the 16MB boundary.

**The extended read-only DSA (ERDSA)**

The key-0 storage area for all reentrant programs and tables above the 16MB boundary.

**The extended shared DSA (ESDSA)**

The user-key storage area for any non-reentrant user-key RMODE(ANY) programs, and also for any storage obtained by programs issuing CICS GETMAIN commands for storage above the 16MB boundary with the SHARED option.

**The extended CICS DSA (ECDSA).**

The CICS-key storage area for all non-reentrant CICS-key RMODE(ANY) programs, all CICS-key task-lifetime storage above the 16MB boundary, and CICS control blocks that reside above the 16MB boundary.

CICS allocates all the DSAs above the 16MB boundary in multiples of 1MB.

**Notes:**

1. For information about calculating the amount of storage to specify on the EDSALIM parameter, see the *CICS Performance Guide*
2. Dynamic changes to the EDSA limit are cataloged in the *local* catalog, and override the DSALIM parameter, if it is specified in the system initialization table, during all forms of restart-initial, cold, and warm. The cataloged value is *not* used if:
  - You specify startup values as system initialization parameters overrides (for example, in SYSIN).
  - You re-initialize the CICS catalog data sets.

**EJBROLEPRFX=*ejbrole-prefix***

Specifies a prefix that is used to qualify the security role defined in an enterprise bean's deployment descriptor. The prefix is applied to the security role:

- when a role is defined to an external security manager
- when CICS maps a security role to a RACF user ID
- when an application invokes the `isCallerInRole()` method

For more information about how the EJBROLEPRFX parameter is used to qualify security roles for enterprise beans, see *Java Applications in CICS*.

You can specify a prefix of up to 16 characters. The prefix must not contain a period (.) character. If you specify a prefix that contains lower case characters,

blanks, or punctuation characters, you must enclose it in apostrophes. If the prefix contains an apostrophe, code two successive apostrophes to represent it.

**Restrictions:**

1. You can specify the EJBROLEPRFX parameter in the SIT, PARM, or SYSIN only.
2. The EJBROLEPRFX parameter is ignored if security role support is not enabled. To enable security role support you must specify SEC=YES and XEJB=YES.

**ENCRYPTION={ STRONG|WEAK|MEDIUM}**

Specifies the cipher suites that CICS uses for secure TCP/IP connections. When a secure connection is established between a pair of processes, the most secure cipher suite supported by both is used.

- Use ENCRYPTION=STRONG when you can tolerate the overhead of using high encryption if the other system requires it.
- Use ENCRYPTION=WEAK when you want to use encryption up to 40 bits in length.
- Use ENCRYPTION=MEDIUM when you want to use encryption up to 56 bits in length.

For compatibility with previous releases, ENCRYPTION=NORMAL is accepted as an equivalent to ENCRYPTION=MEDIUM. For more information about cipher suites, see *CICS RACF Security Guide*.

CICS can use only the cipher suites that are supported by the underlying z/OS operating system. For an up to date list of the cipher suites that are supported, check the appropriate z/OS documentation.

Possible values for z/OS 1.9 are:

**STRONG**

Specifies that CICS should use the following cipher suites:

Cipher suite	Encryption algorithm	Key length	MAC algorithm	Key exchange
01	No encryption		MD5	None
02	No encryption		SHA-1	None
03	RC4	40 bits	MD5	RSA
04	RC4	128 bits	MD5	RSA
05	RC4	128 bits	SHA-1	RSA
06	RC2	40 bits	MD5	RSA
09	DES	56 bits	SHA-1	RSA
0A	3DES	168 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate
0C	DES	56 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate

<b>Cipher suite</b>	<b>Encryption algorithm</b>	<b>Key length</b>	<b>MAC algorithm</b>	<b>Key exchange</b>
0D	3DES	168 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate
0F	DES	56 bits	SHA-1	Fixed Diffie-Hellman key exchange using RSA certificate
10	3DES	168 bits	SHA-1	Fixed Diffie-Hellman key exchange using RSA certificate
12	DES	56 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using DSS certificate
13	3DES	168 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using DSS certificate
15	DES	56 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using RSA certificate
16	3DES	168 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using RSA certificate
2F	AES	128 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate
30	AES	128 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate
31	AES	128 bits	SHA-1	Fixed Diffie-Hellman key exchange using RSA certificate

Cipher suite	Encryption algorithm	Key length	MAC algorithm	Key exchange
32	AES	128 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using DSS certificate
33	AES	128 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using RSA certificate
35	AES	256 bits	SHA-1	RSA
36	AES	256 bits	SHA-1	Fixed Diffie-Hellman key exchange using DSS certificate
37	AES	256 bits	SHA-1	Fixed Diffie-Hellman key exchange using RSA certificate
38	AES	256 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using DSS certificate
39	AES	256 bits	SHA-1	Ephemeral Diffie-Hellman key exchange using DSS certificate
The terms used in this table are:				
<b>AES</b>	Advanced Encryption Standard			
<b>DES</b>	Data Encryption Standard			
<b>DSS</b>	Digital Signature Standard			
<b>MD5</b>	Message Digest algorithm			
<b>RC2, RC4</b>	Rivest encryption			
<b>RSA</b>	Rivest-Shamir-Adleman encryption			
<b>SHA-1</b>	Secure Hash algorithm			
<b>3DES</b>	DES applied three times			

#### **WEAK**

Specifies that CICS should use the following cipher suites:

Cipher suite	Encryption algorithm	Key length	MAC algorithm
01	No encryption		MD5
02	No encryption		SHA

Cipher suite	Encryption algorithm	Key length	MAC algorithm
03	RC4	40 bits	MD5
06	RC2	40 bits	MD5
The terms used in this table are:			
<b>MD5</b>	Message Digest algorithm		
<b>SHA</b>	Secure Hash algorithm		
<b>RC2, RC4</b>	Rivest encryption		

## MEDIUM

Specifies that CICS should use the following cipher suites:

Cipher suite	Encryption algorithm	Key length	MAC algorithm
01	No encryption		MD5
02	No encryption		SHA
03	RC4	40 bits	MD5
06	RC2	40 bits	MD5
09	DES	56 bits	SHA
The terms used in this table are:			
<b>MD5</b>	Message Digest algorithm		
<b>SHA</b>	Secure Hash algorithm		
<b>RC2, RC4</b>	Rivest encryption		
<b>DES</b>	Data Encryption Standard		

**EODI**={E0|xx}

specifies the end-of-data indicator for input from sequential devices. The characters "xx" represent two hexadecimal digits in the range 01 through FF. The default value is X'E0', which represents the standard EBCDIC backslash symbol (\).

**ERDSASZE**={0K|number}

specifies the size of the ERDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

### number

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the ERDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**ESDSASZE**={0K|number}

specifies the size of the ESDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the ESDSAZSE parameter in PARM, SYSIN, or CONSOLE only.

**ESMEXITS={NOINSTLN|INSTLN}**

specifies whether installation data is to be passed through the RACROUTE interface to the external security manager (ESM) for use in exits written for the ESM.

**NOINSTLN**

The INSTLN parameter is not used in RACROUTE macros.

**INSTLN**

CICS-related and installation-supplied data is passed to the ESM using the INSTLN parameter of the RACROUTE macro. For programming information, including the format of the data passed, see the *CICS Customization Guide*. This data is intended for use in exits written for the ESM.

**Restrictions** You can specify the ESMEXITS parameter in the SIT only.

**EUDSAZSE={0K|number}**

specifies the size of the EUDSA. The default size is 0 indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

Specify number as an amount of storage in the range 0 to 1073741824 bytes in multiples of 1048576 bytes (1MB). If the size specified is not a multiple of 1MB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the EUDSAZSE parameter in PARM, SYSIN, or CONSOLE only.

**FCT={NO|xx|YES}**

specifies the suffix of the file control table to be used.

This parameter is effective only on a CICS cold or initial start. CICS does not load an FCT on a warm or emergency restart, and all file resource definitions are recovered from the global catalog.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

You can use a mixture of macro definitions and RDO definitions for files in your CICS region. However, your FCT should contain definitions for only BDAM files to be loaded on a CICS cold start. Other types of files are loaded from their file definitions in RDO groups specified in the GRPLIST system initialization parameter. Any definitions in the FCT other than for BDAM files are ignored.

**FEPI**={NO|YES}

specifies whether or not you want to use the Front End Programming Interface feature (FEPI).

**NO** FEPI support is not required. You should specify NO on this parameter (or allow it to default) if you do not have the feature installed, or if you do not require FEPI support.

**YES** You require FEPI support, and CICS is to start the CSZI transaction.

This book does not contain any information about the installation process for the Front End Programming Interface feature. Installation information can be found in the *CICS Front End Programming Interface User's Guide*.

**FLDSEP**={' ' | 'xxxx'

specifies one through four field-separator characters, each of which indicates end of field in the terminal input data. The default is four blanks.

The field separator allows you to use transaction identifications of less than four characters followed by one of the separator characters. When less than four characters are coded, the parameter is padded with blanks, so that the blank is then a field separator. None of the specified field separator characters should be part of a transaction identification; in particular, the use of alphabetic characters as field separators is not recommended.

The character specified in the FLDSEP parameter must not be the same as any character specified in the FLDSTRT parameter. This means that it is invalid to allow both parameters to take the default value. **Restrictions**

If you specify FLDSEP in the SIT, the characters must be enclosed in single quotation marks.

If you specify FLDSEP as a PARM, SYSIN, or CONSOLE parameter, do **not** enclose the characters in quotation marks, and the characters you choose must not include an embedded blank, or any of these characters:

( ) ' = ,

**FLDSTRT**={' ' | 'x'}

specifies a single character to be the field-name-start character for free-form input for built-in functions. The default is a blank.

The character specified should not be part of a transaction identification; in particular, the use of alphabetic characters is not recommended.

The character specified in the FLDSTRT parameter must not be the same as any character specified in the FLDSEP parameter. This means that it is invalid to allow both parameters to take the default value.

#### **Restrictions**

If you specify FLDSTRT in the SIT, the parameter must be enclosed in single quotation marks.

If you specify FLDSTRT as a PARM, SYSIN, or CONSOLE parameter, do **not** enclose the character in quotation marks, and the character you choose must not be a blank or any of the following characters:

( ) ' = ,

### **FORCEQR={NO|YES}**

Specifies whether you want CICS to force all CICSAPI user application programs that are specified as threadsafe to run under the CICS QR TCB, as if they were specified as quasi-reentrant programs. This parameter applies to all application programs that are restricted to the current CICS programming interfaces ( that is, those which specify API(CICSAPI) ), and does not apply to any of the following:

- Java programs that are run in a JVM,
- C/C++ programs using XPLINK,
- OPENAPI programs,

none of which can run on the QR TCB.

**NO** CICS is to honor the CONCURRENCY(THREADSAFE) attribute on program resource definitions, and allow user application programs to run on an open TCB to avoid unnecessary TCB switching.

**YES** CICS is to force all CICSAPI user application programs specified with the CONCURRENCY(THREADSAFE) attribute to run under the CICS QR TCB, as if they were specified as CONCURRENCY(QUASIRENT) programs

FORCEQR=YES will allow you, in a test environment, to run incompletely tested threadsafe application programs that have proved to be non-threadsafe.

FORCEQR will apply to all programs defined as threadsafe that are not invoked as task-related user exits, global user exits, or user-replaceable modules.

### **FSSTAFF={YES|NO}**

specify this parameter in an application-owning region (AOR) to prevent transactions initiated by function-shipped EXEC CICS START requests being started against incorrect terminals.

You may need to code the function-shipped START affinity (FSSTAFF) parameter in an AOR if all of the following are true:

1. The AOR is connected to two or more terminal-owning regions (TORs) that use the same, or a similar, set of terminal identifiers.
2. One or more of the TORs issues EXEC CICS START requests for transactions in the AOR.
3. The START requests are associated with terminals.
4. You are using shippable terminals, rather than statically defining remote terminals in the AOR.

Consider the following scenario:

Terminal-owning region TOR1 issues an EXEC CICS START request for transaction TRAR, which is owned by region AOR1. It is to be run against terminal T001. Meanwhile, terminal T001 on region TOR2 has been transaction routing to AOR1; a definition of T001 has been shipped to AOR1 from TOR2. When the START request arrives at AOR1, it is shipped to TOR2, rather than TOR1, for transaction routing from terminal T001.

To prevent this situation, code YES on the FSSTAFF parameter in the AOR.

**YES** When a START request is received from a terminal-owning region, and a shipped definition for the terminal named on the request is already installed in the AOR, the request is always shipped back to a TOR, for routing, *across the link it was received on*, irrespective of the TOR referenced in the remote terminal definition.

If the TOR to which the START request is returned is **not** the one referenced in the installed remote terminal definition, a definition of the terminal is shipped to the AOR, and the autoinstall user program is called. Your autoinstall user program can then allocate an *alias* termid in the AOR, to avoid a conflict with the previously installed remote definition. For information about writing an autoinstall program to control the installation of shipped definitions, see the *CICS Customization Guide*.

**NO** When a START request is received from a terminal-owning region, and a shipped definition for the named terminal is already installed in the AOR, the request is shipped to the TOR referenced in the definition, for routing.

**Notes:**

1. FSSTAFF has no effect:
  - On statically-defined (hard-coded) remote terminal definitions in the AOR. If you use these, START requests are always shipped to the TORs referenced in the definitions.
  - On START requests issued in the local region. It affects only START requests shipped from other regions.
  - When coded on intermediate regions in a transaction-routing path. It is effective only when coded on an application-owning region.
2. If the AOR contains no remote definition of a terminal named on a shipped START request, the “terminal not known” global user exits, XICTENF and XALTENF, are called. For details of these exits, see the *CICS Customization Guide*.

**FTIMEOUT={30|nn}**

specifies a timeout interval for requests made on files that are opened in RLS mode. The interval is in seconds, from 1 through 4080 (sixty eight minutes) and indicates how long VSAM should wait before terminating a request and returning an exception condition.

The default is 30 seconds.

FTIMEOUT applies to transactions that do not have a deadlock timeout interval active. If the DTIMOUT keyword of the TRANSACTION definition is specified, it is used as the file timeout value for that transaction.

**GMTEXT={'DFHZC2312 \*\*\* WELCOME TO CICS \*\*\*'|'text'}**

specifies whether the default logon message text (WELCOME TO CICS) or your own message text is to be displayed on the screen by the CSGM (good morning) transaction when a terminal is logged on to CICS through VTAM, by the CESN transaction if used to sign on to CICS, or by your own transactions using the EXEC CICS INQUIRE SYSTEM GMMTEXT command.

You can use apostrophes to punctuate your message, in addition to using them as message delimiters. However, you must code *two* successive apostrophes to represent a single apostrophe in your text. For example,

```
GMTEXT='User''s logon message text.'
```

The whole message must still be enclosed by a pair of single delimiting apostrophes.

Your message text can be from 1 through 246 characters (bytes), and can extend over two lines by extending the text to column 80 on the first line, and continuing in column 1 of the second line. For example, the following might be used in the SYSIN data set:

```
*          CICS Transaction Server for z/OS, Version 3 Release 1 SYSTEM  *
GMTEXT='An Information Development CICS Terminal-Owning Region (TOR) - C
ICSIDC. This message is to show the use of continuation lines when creating a GM
TEXT parameter in the SYSIN data set' (for first signon
```

The CSGM transaction displays this as follows (with the time appended to the end of message):

```
An Information Development CICS Terminal-Owning Region (TOR) - C
ICSIDC. This message is to show the use of continuation lines when creating a GM
TEXT parameter in the SYSIN data set 09:56:14
```

The CESN transaction displays this as follows:

```
Signon for CICS Transaction Server for z/OS, Version 3 Release 1 APPLID CICSHTH1
An Information Development CICS Terminal-Owning Region (TOR) - CICSIDC.
This message is to show the use of continuation lines when creating a GMTEXT
parameter in the SYSIN data set
```

For any transaction other than CESN that displays the text specified by this parameter, you must use a TYPETERM with LOGONMSG(YES) for all terminals requiring the logon message. For information about using TYPETERM, see the *CICS Resource Definition Guide*.

**GMTRAN={CSGM|CESN|transaction-id}**

specifies the name of the transaction that is:

1. initiated by ATI when terminals are logged on to CICS by VTAM, and LOGONMSG(YES) is specified in the TYPETERM definition.
2. set to be the next transaction initiated by the terminal operator following expiry of the terminal user's TIMEOUT period (specified in the External Security Manager) and either:
  - LOGONMSG(YES) and SIGNOFF(YES)

or

- LOGONMSG(YES), SIGNOFF(LOGOFF) and DISCREQ(NO)

is specified in the TYPETERM definition.

§ initiated when terminals are logged on to CICS by VTAM. Do not specify the name of a remote transaction. The transaction must be capable of being automatically initiated (ATI). The default is the transaction CSGM, that displays the text specified in the GMTEXT parameter. Alternatively, you can specify the CICS signon transaction, CESN, which also displays the text specified in the GMTEXT parameter. The GMTRAN parameter can be used with the LGNMSG parameter to retrieve VTAM logon data.

**GNTRAN={NO|transaction\_id}**

specifies the transaction that you want CICS to invoke when a user's terminal-timeout period expires.

**NO** The default value, NO, specifies that no special transaction is to be executed when the timeout period expires. Instead, the user is signed off (subject to the SIGNOFF attribute of the TYPETERM resource definition for the terminal, as described below). After the signoff, if the

LOGONMSG(YES) option is specified in the TYPETERM resource definition for the terminal, the transaction specified in the GMTRAN system initialization parameter is executed.

#### **transaction\_id**

The name of a timeout transaction to signoff the user at the timed-out terminal. You can specify CESF as the timeout transaction. Specifying your own transaction allows you to specify functions in addition to, or instead of, signoff. For example, your own transaction could issue a prompt for the terminal user's password, and allow the session to continue if the correct password is entered.

The transaction to be used must have been specially written to handle the GNTRAN commarea that is passed to it. Of the CICS-supplied transactions, only CESF has been written to handle the GNTRAN commarea. For more information about writing your own transactions for GNTRAN, see the *CICS Customization Guide*.

**Note:** When either the CICS CESF transaction, or your own transaction, attempts to sign off a terminal, the result is subject to the SIGNOFF attribute of the TYPETERM resource definition for the terminal, as follows:

<b>SIGNOFF</b>	<b>Effect</b>
<b>YES</b>	The terminal is signed off, but not logged off.
<b>NO</b>	The terminal remains signed on and logged on.
<b>LOGOFF</b>	The terminal is both signed off and logged off.

**Note:** If GNTRAN fails to attach, and SIGNOFF(LOGOFF) has been specified, the terminal which has reached timeout will be signed off and logged off. GNTRAN will not run and will have no effect.

#### **GRNAME=name**

specifies the VTAM generic resource name, as 1 through 8 characters, under which a group of CICS terminal-owning regions in a CICSplex register to VTAM.

There is no default for GRNAME. If you do not specify GRNAME, CICS does not register itself with the VTAM generic resources function. Do not confuse the term *generic applid* with *generic resource name*. Generic applids apply only to CICS regions that use XRF. Generic resource names apply only to VTAM generic resource groups.

#### **Notes:**

1. If you are operating a CICSplex that comprises separate terminal-owning regions and application-owning regions, you should ensure that you define a VTAM generic resource name to the CICS terminal-owning regions only.
2. If you specify the XRF=YES parameter, you should not specify a value for the GRNAME system initialization parameter. Any value specified for GRNAME is set to blanks.
3. If you specify the XRF=NO parameter, and a value for GRNAME, you should not specify a specific applid (name2) for the APPLID system initialization parameter. Any specific applid (name2) specified for the APPLID parameter is used for the generic applid (name1); that is, the CICS region will be known to VTAM by the value of the specific applid.
4. Generic resource names must be unique within a single network. A generic resource cannot be identical to:
  - A USERVAR
  - An alias name

- A real LU name

**Note:** It is the responsibility of the user to see that these rules are kept.

5. The first character of the GRNAME cannot be a number.

For example, a CICS region with the system initialization parameters:

```
APPLID=CICSHTH1
GRNAME=CICSH###
```

would register to VTAM with the applid CICSHTH1 and the generic resource CICSH###. Other LUs in the same sysplex can communicate with the CICS region either through the generic resource or the applid.

The examples used here are based on a CICS naming convention described in the *MVS Sysplex Application Migration* manual.

However, care should be taken with LU6 connections initiated from this side (such as AUTOCONNECT(YES)) because the bind will now contain the generic resource name and may fail if the partner only knows this region by the applid. Binds initiated from the partner are examined to identify the name by which the partner knows this region (generic resource or applid), thus allowing the appropriate connection to be built. Recommendations on defining connections can be found in the *CICS Intercommunication Guide*.

**Note:** There are rules that restrict CICS use of the VTAM generic resources function; for more information see the *CICS Intercommunication Guide*.

**GRPLIST={DFHLIST | name | (name [, name2] [, name3] [, name4]) }**

specifies the names (each 1 through 8 characters) of up to four lists of resource definition groups on the CICS system definition (CSD) file. The resource definitions in all the groups in the specified lists are loaded during initialization when CICS performs a cold start. If a warm or emergency start is performed, the resource definitions are derived from the global catalog, and the GRPLIST parameter is ignored.

Each name can be either a real group list name or a generic group list name that incorporates global filename characters (+ and \*). If you specify more than one group list (either by specifically coding two or more group list names or by coding a group list name with global filename characters), the later group lists are concatenated onto the first group list. Any duplicate resource definitions in later group lists override those in earlier group lists.

Use the CEDA command LOCK to protect the lists of resource groups specified on the GRPLIST parameter.

The default is DFHLIST, the CICS-supplied list that specifies the set of resource definitions needed by CICS. If you create your own group list, either add to it the groups specified in DFHLIST (omitting only those for CICS functions that you know you do not need) or specify the DFHLIST name on the GRPLIST parameter. Do not code GRPLIST=NO unless you have a group list named NO.

**Notes:**

1. Group lists specified by a generic group list name are concatenated in alphabetic then numeric order. For example, the generic list name CICSHT\* would concatenate the group lists CICSHT#1, CICSHTAP, CICSSD, and CICSHT3V in that order. If the order of concatenation is important (for example, to ensure that a particular resource definition overrides another), you should consider coding real group list names.

2. If a group list contains resource definitions that are needed by another group list, the group list containing those definitions must be installed first. For example, if list A has TYPETERM definitions needed for TERMINAL definitions in list B, list A must be installed first. This may mean that you have to specifically name the prerequisite group on the GRPLIST parameter.
3. Take care when using generic group list names, because if a group list on your CSD satisfies the generic name, it will be installed. This means that a group list can be installed more than once; for example, if you specify the real group list name and a generic group list name that it satisfies, or if you specify two generic group list names that the group list name satisfies.
4. To override one or more of the group lists specified on the GRPLIST system initialization parameter, you must specify all list names (both real and generic) that you want to use, even if you are not changing the names.

For example, if you want to use the four group lists CICSHT#1, CICSHTAP, CICSHT3V, and CICSHTSD, you could specify either of the following system initialization parameters:

```
GRPLIST=(CICSHT#1,CICSHTAP,CICSHT3V,CICSHTSD)
GRPLIST=(CICSHT*)
```

In the first example GRPLIST, the group lists are loaded in the order specified, and resource definitions installed from the CICSHTSD group list will override any duplicate definitions installed by the other groups.

In the second example GRPLIST, the group lists are loaded in the order CICSHT#1, CICSHTAP, CICSHTSD, then CICSHT3V, and resource definitions installed from the CICSHT3V group list will override any duplicate definitions installed by the other groups.

If your SIT contains the parameter:

```
GRPLIST=(CICSHT#1,CICSAP*,CICSHT3V,CICSHTSD)
```

and you want to replace the list CICSHT3V with the list ANOLST05, you should specify the override:

```
GRPLIST=(CICSHT#1,CICSAP*,ANOLST05,CICSHTSD)
```

In general, any required resource definitions should appear in *one of* the group lists specified on the GRPLIST system initialization parameter.

For information about resource definitions, groups, lists, and the CSD, see the *CICS Resource Definition Guide*.

#### **GTFTTR={OFF|ON}**

specifies whether CICS can use the MVS generalized trace facility (GTF) as a destination for trace data.

This parameter controls whether any of the three types of CICS trace entry are written to GTF data sets. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (which are always made and not controlled by a system initialization parameter).

**OFF** CICS does not use GTF as a destination for CICS trace data.

**ON** CICS uses GTF as a destination for CICS trace data. To use the GTF data sets for CICS trace data, you must have started GTF with the USR option, in addition to coding GTFTTR=ON.

For information about GTF, see the *OS/390 MVS Diagnosis: Tools and Service Aids* manual, SY28-1985.

**HPO={NO|YES}**

specifies whether you want to use the VTAM authorized path feature of the high performance option (HPO). If you code YES, the CICS type 6 SVC must be link-edited in your MVS nucleus, and defined to MVS in an SVC Parm statement. If the SVC number is not 215 (the default) you must specify the SVC number on the SRBSVC parameter.

For information about installing the CICS type 6 SVC in your MVS system, and about changing the default number, see the *CICS Transaction Server for z/OS Installation Guide*.

**Restrictions** You can specify the HPO parameter in the system initialization table only.

**ICP=COLD**

specifies that you want to cold start the interval control program. See page "Defining CICS resource table and module keywords" on page 162 for further information. If COLD is not specified, the ICP start type will be determined by the START and TS parameter values.

**ICV={1000|number}**

specifies the region exit time interval in milliseconds. The ICV system initialization parameter specifies the maximum time in milliseconds that CICS releases control to the operating system when there are no transactions ready to resume processing. This time interval can be any integer in the range 100 through 3600000 milliseconds (specifying an interval up to 60 minutes). A typical range of operation might be 100 through 2000 milliseconds.

A low value interval can enable much of the CICS nucleus to be retained in dynamic storage, and not be paged-out at times of low terminal activity. This reduces the amount of dynamic storage paging necessary for CICS to process terminal transactions (thus representing a potential reduction in response time), sometimes at the expense of concurrent batch region throughput. Large networks with high terminal activity are inclined to run CICS without a need for this value, except to handle the occasional, but unpredictable, period of inactivity. These networks can usually function with a large interval (10000 to 3600000 milliseconds). Once a task has been initiated, its requests for terminal services and the completion of the services are recognized by the system and this maximum delay interval is overridden.

Small systems, or those with low terminal activity, are subject to paging introduced by other jobs running in competition with CICS. By specifying a low value interval, key portions of the CICS nucleus are referenced more frequently, thus reducing the probability of these pages being paged-out. However, the execution of the logic without performing productive work might be considered wasteful. The need to increase the probability of residency by frequent but unproductive referencing must be weighed against the overhead and response time degradation incurred by allowing the paging to occur. By increasing the interval size, less unproductive work is performed at the expense of performance if paging occurs during the periods of CICS activity. For information about the effect of ICV on performance, see the *CICS Performance Guide*.

**Note:** The region exit time interval process contains a mechanism to ensure that CICS does not constantly set and cancel timers (thus degrading performance) while attempting to meet its objectives for a low region exit time interval. This mechanism can cause CICS to release control to the operating system for up to 0.5 seconds when the interval has been set at

less than 250; and up to 0.25 seconds more than the region exit time interval when the interval has been set greater than 250.

**ICVR={5000|number}**

specifies the default runaway task time interval in milliseconds as a decimal number. You can specify zero, or a number in the range 500 through 2 700 000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval used by transactions defined with RUNAWAY=SYSTEM (see the *CICS Resource Definition Guide* for further information). CICS may purge a task if it has not given up control after the RUNAWAY interval for the transaction (or ICVR if the transaction definition specified RUNAWAY=SYSTEM). If you code ICVR=0, runaway task control is inoperative for transactions specifying RUNAWAY=SYSTEM in their transaction definition (that is, tasks do not get purged if they appear to be looping). The ICVR value is independent of the ICV value, and can be less than the ICV value. Note that CICS runaway task detection is based upon task time, that is, the interval is decremented only when the task has control of the processor. For information about commands that reinitialize the ICVR value, see the *CICS Problem Determination Guide*.

**ICVTSD={500|number}**

specifies the terminal scan delay value. The terminal scan delay facility determines how quickly CICS deals with some terminal I/O requests made by applications. The range is 0 through 5000 milliseconds, with a default of ICVTSD=500.

There is an overhead in dealing with such requests. By specifying a nonzero value, the overhead may be spread over several transactions. A value close to zero (for example 200) would be adequate.

**IIOPLISTENER={YES|NO}**

specifies whether the CICS region is to function as an IIOPLISTENER region.

**YES** The CICS region is an IIOPLISTENER region, or a combined listener and application owning region (AOR).

**NO** The CICS region is an IIOPLISTENER application owning region, and TCPIP SERVICE definitions installed in the region, that specify PROTOCOL(IIOPLISTENER), cannot be opened.

This parameter has no effect if the region is not an IIOPLISTENER region or an AOR.

For more information about IIOPLISTENER regions and AORs, see *Java Applications in CICS*

**INFOCENTER=servername:portnumber**

Specifies the server name of where the CICS Information Center is installed, and the port number that it uses to run in server mode. The port number is specified in the start up script for the information center. The default value is 29127, but you can change it to a suitable number by editing the script file. CICS-supplied transactions with a Web browser interface use the value of this parameter to construct links to topics in the information center.

**Example:** INFOCENTER=http://server\_name:29127

If you do not code this parameter, CICS does not construct links to the information center.

**INITPARM=(pgmname\_1='parmstring\_1'[, .... ,pgmname\_n='parmstring\_n'])**

specifies that parameters are to be passed to application programs that use the

ASSIGN INITPARM command. For example, you can use INITPARM to pass parameters to PLTPI programs to be executed in the final stages of system initialization. The area giving access to the parameters is specified by the ASSIGN INITPARM command. For programming information about the ASSIGN INITPARM command, see the *CICS Application Programming Reference*.

**pgmname**

The name of a program. This name must be 1 through 8 alphanumeric or national language characters.

**parmstring**

The parameter string (up to 60 characters enclosed by single quotes) to be passed to the associated program. Any quotes imbedded in the string must be duplicated. For information on coding INITPARM in the SYSIN data set, see “Rules for coding CICS system initialization parameters in the SYSIN data set” on page 277.

You can specify up to 255 pgmname='parmstring' sets.

**Note:** You can specify the INITPARM keyword and its parameters more than once, see “Sample startup job stream, note 5” on page 344.

**INTTR={ON|OFF}**

specifies whether the internal CICS trace destination is to be activated at system initialization.

This parameter controls whether any of the three types of CICS trace entry are written to the internal trace table. The three types are: CICS system trace (see the SYSTR parameter), user trace (see the USERTR parameter), and exception trace entries (which are always made and not controlled by a system initialization parameter).

**ON**     Activate main storage trace.

**OFF**    Do not activate main storage trace.

**IRCSTRT={NO|YES}**

specifies whether IRC is to be started up at system initialization. If IRCSTRT=YES is not coded, IRC can be initialized by issuing a CEMT or EXEC CICS SET IRC OPEN command.

**ISC={NO|YES}**

specifies whether the CICS programs required for interregion or intersystem communication are to be included.

**JESDI={30|number} (alternate)**

specifies, in a SIT for an alternate XRF system, the JES delay interval, in seconds, the minimum being 5 seconds. The alternate CICS region has to ensure that the active CICS region has been canceled before it can take over the resources owned by the active.

**Note:** You must give careful consideration to the values you specify for the parameters ADI and JESDI so that they do not conflict with your installation's policy on PR/SM RESETTIME and the XCF INTERVAL and OPNOTIFY intervals. You should ensure that the sum of the interval you specify for ADI plus JESDI exceeds the interval specified by the XCF INTERVAL and the PR/SM policy interval RESETTIME.

**JVMCCPROFILE={DFHJVMCC|profile}**

specifies the JVM profile to be used for the master JVM that initializes the shared class cache. The default is the supplied sample JVM profile,

DFHJVMCC, which you can modify. The value specified by JVMCCPROFILE is used on an initial or cold start of CICS. You can restart the shared class cache with a different JVM profile while CICS is running. On subsequent restarts, the value from the last CICS execution is used, unless you provide JVMCCPROFILE as a SIT override.

When you specify the JVM profile, use the same combination of upper and lower case characters that is present in the HFS file name of the JVM profile.

**JVMCCSIZE={24M|number}**

specifies the size of the shared class cache on an initial or cold start of CICS. The size of the shared class cache can be between 1MB and 2047MB. You can specify the number in bytes, or as a whole number of kilobytes followed by the letter K, or as a whole number of megabytes followed by the letter M. The default value is 24MB (specified as 24M). You can use the CEMT PERFORM CLASSCACHE START or RELOAD command (or the equivalent EXEC CICS command) to change the size of the shared class cache while CICS is running. On subsequent restarts, the value from the last CICS execution is used, unless you provide JVMCCSIZE as a SIT override.

**JVMCCSTART={AUTO|YES|NO}**

determines whether or not the shared class cache is started during CICS initialization, and sets the status of autostart for the shared class cache. When autostart is enabled for the shared class cache, if the shared class cache has been stopped or has not yet been started, it is started as soon as CICS receives a request to run a Java application in a JVM whose profile requires the use of the shared class cache. When autostart is disabled, the shared class cache can only be started by a CEMT PERFORM CLASSCACHE START command (or the equivalent EXEC CICS command). You can change the status of autostart while CICS is running, either by using the AUTOSTARTST option on the CEMT PERFORM CLASSCACHE command, or by using the CEMT SET CLASSCACHE command (or the equivalent EXEC CICS commands). If you do this, subsequent CICS restarts use the changed setting, unless the system is INITIAL or COLD started, or the JVMCCSTART system initialization parameter is specified as an override at startup. In these cases, the setting from the system initialization parameter is used.

**AUTO**

The shared class cache is not started during CICS initialization. Autostart is enabled, so the shared class cache will start as soon as a JVM needs it.

**YES**

The shared class cache is started during CICS initialization. Autostart is enabled, so if you stop the shared class cache, it is restarted as soon as a JVM needs it.

**NO**

The shared class cache is not started during CICS initialization. Autostart is disabled, so the shared class cache will not start unless you issue a CEMT PERFORM CLASSCACHE START command (or the equivalent EXEC CICS command).

**JVMxxxxTRACE (JVMLEVEL0TRACE=option, JVMLEVEL1TRACE=option, JVMLEVEL2TRACE=option, JVMUSERTRACE=option)**

These system initialization parameters specify the default options for JVM tracing. "Defining tracing for JVMs" in the *CICS Problem Determination Guide* has information about the JVM trace options that you can set using the JVMLEVEL0TRACE, JVMLEVEL1TRACE, JVMLEVEL2TRACE and JVMUSERTRACE system initialization parameters. There is further information about JVM trace and about problem determination for JVMs in the *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/](http://www.ibm.com/developerworks/java/jdk/diagnosis/).*

Trace levels 29–32 for the SJ component correspond to JVMLEVEL0TRACE, JVMLEVEL1TRACE, JVMLEVEL2TRACE and JVMUSERTRACE respectively. To activate JVM tracing, specify level numbers 29–32 on the SPCTRSJ or STNTRSJ system initialization parameter, or use the CETR transaction.

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.

For definitions of the individual JVM tracing parameters, see “JVMLEVEL0TRACE,” “JVMLEVEL1TRACE,” “JVMLEVEL2TRACE,” and “JVMUSERTRACE” below.

**Restrictions:** You can specify the JVMxxxxTRACE parameters in PARM, SYSIN, or CONSOLE only.

- # **JVMLEVEL0TRACE={'ALL (EXCEPTION)' | 'user override string'}**  
Specifies the default option for JVM Level 0 trace, corresponding to trace level 29 of the SJ component. The default setting for this level of tracing maps to trace point level 0 for JVMs, which is reserved for extraordinary events and errors. Unlike CICS exception trace, which cannot be switched off, the JVM Level 0 trace is normally switched off unless JVM tracing is required. “JVMxxxxTRACE” on page 206 has more information about these system initialization parameters.
- # **JVMLEVEL1TRACE={'ALL (ENTRY, EXIT)' | 'user override string'}**  
Specifies the default option for JVM Level 1 trace, corresponding to trace level 30 of the SJ component. The default setting for this level of tracing maps to trace point level 1 for JVMs. “JVMxxxxTRACE” on page 206 has more information about these system initialization parameters.
- # **JVMLEVEL2TRACE={'ALL' | 'user override string'}**  
Specifies the default option for JVM Level 2 trace, corresponding to trace level 31 of the SJ component. The default setting for this level of tracing maps to trace point level 2 for JVMs. Note that the JVM trace point levels go up to level 9. “JVMxxxxTRACE” on page 206 has more information about these system initialization parameters.
- # **JVMUSERTRACE={'NONE' | 'user override string'}**  
Specifies the default option for JVM user trace, corresponding to trace level 32 of the SJ component. Use this option for more complex specifications for JVM tracing. “JVMxxxxTRACE” on page 206 has more information about these system initialization parameters.
- JVMPROFILEDIR={/usr/lpp/cicsts/cicsts31/JVMProfiles | directory}**  
specifies the name (up to 240 characters long) of an HFS directory that contains the JVM profiles for CICS. CICS searches this directory for the profiles it needs to configure JVMs. The default value of JVMPROFILEDIR is /usr/lpp/cicsts/cicsts31/JVMProfiles. That is, the supplied setting for JVMPROFILEDIR points to the **default** directory for the sample JVM profiles. If you chose a different name during CICS installation for the cicsts31 directory beneath which the sample JVM profiles are stored (that is, if you chose a non-default value for the CICS\_DIRECTORY variable used by the DFHIJVMJ job), or if you want CICS to load the JVM profiles from a directory other than the samples directory, you need to do one of the following:
- Change the value of the JVMPROFILEDIR system initialization parameter.

- Link to your JVM profiles from the directory specified by JVMPROFILEDIR, by means of UNIX soft links. (This method enables you to store your JVM profiles in any place in the HFS file system.)

Note that the JVM profiles DFHJVMPR and DFHJVMCD, and their associated JVM properties files, must always be available to CICS. DFHJVMPR is used if a Java program is defined as using a JVM but no JVM profile is specified, and it is used for sample programs. DFHJVMCD is used by CICS-defined programs, including the default request processor program and the program that CICS uses to publish and retract deployed JAR files. Both these JVM profiles must therefore either be present in the directory that is specified by JVMPROFILEDIR, or linked to by means of UNIX soft links from that directory.

**KEYRING**=*keyring-name*

Specifies the fully qualified name of the key ring, within the external security manager's database, that contains the keys and X.509 certificates used by CICS support for the secure sockets layer (SSL) and for Web services security.

Be aware that the key ring name is case sensitive.

**Notes:**

1. The maximum length of the KEYRING parameter is 47 characters.
2. For more information on creating a key ring, see the *CICS RACF Security Guide*

**LGDFINT**={5|*number*

# specifies the log defer interval to be used by CICS log manager when  
# determining how long to delay a forced journal write request before invoking the  
# MVS system logger. The value is specified in milliseconds.

# **5** This is the default.

# **Note:** When this parameter was first introduced, the default value was  
# 30 milliseconds, but customer experience has shown that 5 is a  
# more realistic value.

# **number**

# number can be any value in the range 0 through 65535. You are  
# recommended to allow LGDFINT to assume its default value, 5.

# **Note:** You can modify the log defer interval dynamically using the LOGDEFER  
# option of the CEMT SET SYSTEM or EXEC CICS SET SYSTEM  
# commands. However, you are recommended not to modify this value in a  
# production environment without first performing a system evaluation and  
# performance analysis of any changed value.

# If you change the log defer interval value dynamically, the new value is  
# not cataloged. The log defer interval value is taken from the LGDFINT  
# system initialization parameter in all types of CICS startup.

# When a CICS system has many tasks issuing forced log write requests,  
# these tasks will not be delayed for periods close to the LGDFINT  
# parameter value. This is because a forced log write request is normally  
# issued while a log deferral is already being performed for another task.  
# The actual interval might also be affected by the need for tasks to wait  
# across a partition exit.

**LGNMSG={NO|YES}**

specifies whether VTAM logon data is to be made available to an application program.

**NO** VTAM logon data is not available to an application program.

**YES** VTAM logon data is available to an application program. The data can be retrieved with an EXEC CICS EXTRACT LOGONMSG command. For programming information about this command, see the *CICS Application Programming Reference*.

You can use this parameter with the GMTRAN parameter to retrieve the VTAM logon data at the time a terminal is logged on to CICS by VTAM.

**LLACOPY={YES|NO|NEWCOPY}**

specifies whether CICS is to use the LLACOPY macro or the BLDL macro when locating modules in the DFHRPL concatenation.

**YES** CICS always uses the LLACOPY macro when locating modules in the DFHRPL concatenation.

**NO** CICS always uses the BLDL macro when locating modules in the DFHRPL concatenation.

**NEWCOPY**

CICS uses the LLACOPY only when a NEWCOPY or a PHASEIN is being performed. At all other times, CICS uses the BLDL macro when locating modules in the DFHRPL concatenation.

**Notes:**

1. If you code LLACOPY=NO or LLACOPY=NEWCOPY you can still benefit from having LLA managed data sets within your DFHRPL concatenation. Modules will continue to be loaded from VLF if appropriate.
2. If an LLA managed module has been altered, a BLDL macro may not return the new information and a subsequent load will still return the old copy of the module. To load the new module, an LLACOPY must be issued against that module or a MODIFY LLA,REFRESH command must be issued on a system console.
3. If you set LLACOPY to anything other than NO, ensure that the proper RACF security permissions have been set up first. For more information about this refer to in the *CICS RACF Security Guide*.

**LOCALCCSID={037|CCSID}**

Specifies the default CCSID for the local region.

The CCSID is a value of up to 8 characters. If CCSID value is not specified, the default LOCALCCSID is set to 037. For lists of valid CCSIDs, see:

- "CICS-supported conversions" in the Communicating from CICS on System/390 manual , and
- Appendix F of the *z/OS Support for Unicode: Using Conversion Services* manual SA22-7649 .

**037** the default value for LOCALCCSID.

**CCSID**

represents any other valid CCSID value.

**LPA={NO|YES}**

specifies whether any CICS or user modules can be used from the link pack areas.

**NO** will not load CICS or user modules from the link pack areas.

**YES** CICS or usermodules installed in the LPA or in the ELPA can be used from there, instead of being loaded into the CICS region.

A list of the CICS modules that are read-only, and hence eligible for residence in the link pack areas (LPA or ELPA), are contained in the SMP/E USERMOD supplied on the distribution tape in the CICSTS31.CICS.SDFHSAMP, in a member called DFH\$UMOD. For details of the CICS system initialization parameter PRVMOD that you can use to override LPA=YES for selected modules, see page “PRVMOD” on page 224.

**MAXJVMTCBS**={5|*number*}

specifies the maximum number, in the range 1 through 999, of open TCBs CICS can create in the pool of J8 and J9 mode TCBs for use by Java programs that run in a JVM (the JVM pool). Within this limit, there are no constraints on how many of the TCBs in the JVM pool are J9 TCBs, and how many are J8 TCBs.

The default is 5. The minimum permitted value is 1, meaning that CICS is always able to create at least 1 open TCB for use by a JVM, of either J8 or J9 mode.

JM TCBs, used for the master JVM that initializes the shared class cache, do not count towards the MAXJVMTCBS limit.

“System initialization parameters for open TCBs” on page 152 has more information about managing open TCBs.

**MAXOPENTCBS**={12|*number*}

specifies the maximum number, in the range 1 through 2000, of open TCBs CICS can create in the pool of L8 and L9 mode TCBs. Within this limit, there are no constraints on how many of the TCBs in the pool are L8 TCBs, and how many are L9 TCBs.

- L9 mode TCBs are used for USERKEY OPENAPI application programs.
- L8 mode TCBs are used:
  - for CICSKEY OPENAPI application programs
  - for OPENAPI task related user exits, for example the CICS-DB2 Attachment Facility. (Task related user exits always run in CICSKEY.)
  - and by CICS itself, because CICS uses OPENAPI CICSKEY programs which run on L8 TCBs:
    - when accessing doctemplates and HTTP static responses that are stored on Hierarchical File System (HFS).
    - when processing WebService requests and parsing XML.

The default is 12. The minimum permitted value is 1. “System initialization parameters for open TCBs” on page 152 has more information about managing open TCBs.

**MAXSOCKETS**=*number*

Specifies the maximum number of IP sockets that can be managed by the CICS sockets domain.

If the CICS region userid (the userid under which CICS is running) has superuser authority, the default value is 65535.

If the CICS region userid does not have superuser authority, the maximum possible value is the value of the MAXFILEPROC parameter in SYS1.PARMLIB member BPXPRMxx. If you specify a value greater than this in the MAXSOCKETS system initialization parameter (or by letting CICS use the default), CICS issues a message indicating the value that CICS has used.

#  
#  
|

#  
|

Note that sockets created by Java programs running on threads that are not managed by CICS do not count towards the MAXSOCKETS limit.

| **MAXSSLTCBS={8|number}**

| Specifies the maximum number of S8 TCBs that can run in the SSL pool. The  
| default is 8, but you can specify up to 1024 TCBs.

#

# This value must not exceed the MAXTHREADS and MAXTHREADTASKS  
# parameter values, that are specified in SYS1.PARMLIB member BPXPRMxx.

| **MAXXPTCBS={5|number}**

| specifies the maximum number, in the range 1 through 999, of open X8 and X9  
| TCBs that can exist concurrently in the CICS region. X8 and X9 are the TCBs  
| that are used to provide XPLink support.

| “System initialization parameters for open TCBs” on page 152 has more  
| information about managing open TCBs.

| **MCT={NO|YES|xx}**

| specifies the monitoring control table suffix. (See page “Defining CICS resource  
| table and module keywords” on page 162.) If you specify MCT=NO, CICS  
| monitoring builds dynamically a default MCT, ensuring that default monitoring  
| control table entries are always available for use when monitoring is on and a  
| monitoring class (or classes) is active. You can generate an MCT with a  
| single-character suffix only for use by CICS because single-character suffixes  
| cause an error when the MCT is processed by DFHMNDUP. If you use  
| DFHMNDUP, make sure that you create your MCTs with two-character suffixes.

| For information about coding the macros for this table, see the *CICS Operations  
| and Utilities Guide*.

| **MN={OFF|ON}**

| specifies whether monitoring is to be switched on or off at initialization, and use  
| the individual monitoring class parameters to control which monitoring classes  
| are to be active. (See the MNEVE, MNEXC, and MNPER parameter  
| descriptions.) The default status is that the CICS monitoring facility is **off**. The  
| monitoring status is recorded in the CICS global catalog for use during warm  
| and emergency restarts.

| **OFF** Switch off monitoring.

| **ON** Switch on monitoring. However, unless at least one individual class is  
| active, no monitoring records are written. For details of the effect of  
| monitoring status being on or off, in conjunction with the status of the  
| various monitoring classes, see the following notes:

| **Notes:**

1. If the monitoring status is ON, CICS accumulates monitoring data continuously and, depending on the status of each of the monitoring classes, processes the accumulated data as follows:
  - For the performance and exception monitoring classes, CICS writes the monitoring data for each class that is active to a system management facilities (SMF) data set.
  - For the SYSEVENT monitoring class, CICS notifies the MVS system resources manager (SRM) of the completion of each transaction. This data can be reported using the resource measurement facility (RMF™), or written to SMF data sets, depending on the RMF options in force.For information about the effect of SYSEVENT recording in an MVS workload manager environment, see the *CICS Performance Guide*.

If the monitoring status is OFF, CICS does not accumulate or write any monitoring data, even if any of the monitoring classes are active.

2. You can change the monitoring status and the monitoring class settings at any time, as follows:
  - During a warm restart by coding an MN system initialization parameter in PARM, SYSIN, or through the system console.
  - While CICS is running, by either of:
    - The CEMT SET MONITOR command
    - The EXEC CICS SET MONITOR command

When you change the status of monitoring, the change takes effect immediately. If you change the monitoring status from OFF to ON, monitoring starts to accumulate data and write monitoring records to SMF for all tasks that start after the status change is made **for all active monitoring classes**. If the status is changed from ON to OFF, monitoring stops writing records immediately and does not accumulate monitoring data for any tasks that start after the status change is made.

3. The monitoring status operand can be manipulated independently of the class settings. This means that, even if the monitoring status is OFF, you can change the monitoring class settings and the changes take effect for all tasks that are started after the monitoring status is next set to ON.

For programming information about controlling CICS monitoring, see *CICS System Programming Reference*.

#### **MNCONV={NO|YES}**

specifies whether or not conversational tasks are to have separate performance class records produced for each pair of terminal control I/O requests.

Any clock (including user-defined) that is active at the time such a performance class record is produced is stopped immediately before the record is written. After the record is written, such a clock is reset to zero and restarted. Thus a clock whose activity spans more than one recording interval within the conversational task appears in multiple records, each showing part of the time, and the parts adding up to the total time the clock is active. The high-water-mark fields (which record maximum levels of storage used) are reset to their current values. All other fields are set to X'00', except for the key fields (transid, termid). The monitoring converse status is recorded in the CICS global catalog for use during warm and emergency restarts.

#### **MNEXC={OFF|ON}**

specifies whether the monitoring exception class is to be made active during initialization. The monitoring exception class status is recorded in the CICS global catalog for use during warm and emergency restarts.

**OFF** Set the exception monitoring class to “not active”.

**ON** Set the exception monitoring class to “active”.

For programming information about exception monitoring records, see the *CICS Customization Guide*.

#### **MNFREQ={0|hhmmss}**

specifies the interval for which CICS automatically produces a transaction performance class record for any long-running transaction. The monitoring frequency value is recorded in the CICS global catalog for use during warm and emergency restarts.

**0** No frequency monitoring is active.

**hhmmss**

The interval for which monitoring produces automatically a transaction performance class record for any long-running transaction. Specify a 1 to 6 digit number in the range 001500–240000. Numbers that are fewer than six digits are padded with leading zeroes.

**MNPER={OFF|ON}**

specifies whether the monitoring performance class is to be made active during CICS initialization. The monitoring performance class status is recorded in the CICS global catalog for use during warm and emergency restarts.

**OFF** Set the performance monitoring class to “not active”.

**ON** Set the performance monitoring class to “active”.

For programming information about performance monitoring records, see the *CICS Customization Guide*.

**MNRES={OFF|ON}**

specifies whether transaction resource monitoring is to be made active during CICS initialization. The transaction resource monitoring class status is recorded in the CICS global catalog for use during warm and emergency restarts.

**OFF** Set transaction resource monitoring to not active.

**ON** Set transaction resource monitoring to active.

Transaction resource monitoring applies to CICS file resources when you specify the FILE=*nn* option on the DFHMCT TYPE=INITIAL macro.

For programming information about transaction resource monitoring record formats, see the *CICS Customization Guide*.

**MNSUBSYS={nu11|xxxx}**

specifies the 4-character name to be used as the subsystem identification in the monitoring SYSEVENT class records. If you do not specify a name, the subsystem identification defaults to the first four characters of the *name1* operand of the APPLID system initialization parameter. The monitoring subsystem id is recorded in the CICS global catalog for use during warm and emergency restarts.

For background information on the SYSEVENT class of monitoring data and the subsystem identification, and about the implications for SYSEVENT recording in a MVS Workload Manager environment, see the *CICS Performance Guide*.

**MNSYNC={NO|YES}**

specifies whether you want CICS to produce a transaction performance class record when a transaction takes an implicit or explicit syncpoint (unit-of-work). No action is taken for syncpoint rollbacks. The monitoring syncpoint status is recorded in the CICS global catalog for use during warm and emergency restarts.

**MNTIME={GMT|LOCAL}**

specifies whether you want the time stamp fields in the performance class monitoring data to be returned to an application using the EXEC CICS COLLECT STATISTICS MONITOR(taskno) command in either GMT or local time. The monitoring time value is recorded in the CICS global catalog for use during warm and emergency restarts.

For programming information on the EXEC CICS COLLECT STATISTICS command, see the *CICS System Programming Reference manual*.

**MQCONN={NO|YES}**

specifies whether you want CICS to start the MQSeries® for OS/390 connection

# automatically during initialization. A single CICS address space can be  
# connected to only one queue manager at a time.

**NO** Do not automatically invoke CSQCCODF, the MQSeries attach program, during initialization.

**YES** Invoke the MQSeries attach program, CSQCCODF, automatically during CICS initialization. The other information CICS needs for starting the attachment, such as the MQSeries queue manager subsystem name, is taken from the CSQCPARM operand of an INITPARM system initialization parameter.

Specifying MQCONN=YES means you don't need to define the MQSeries attach program in the CICS post-initialization program list table (PLT).

**Note:** The MQCONN parameter works only if you are using the MQSeries-supplied program, CSQCCODF, to start the CICS-MQSeries connection. MQCONN will not work with your own-written attach program if it has a different name.

For more information about starting a connection to an MQSeries queue manager, see *MQSeries for MVS/ESA: System Management Guide*, SC33-0806.

**MROBTCH={1|number}**

specifies the number of events that must occur before CICS is posted for dispatch due to the batching mechanism. The number can be in the range 1 through 255, and the default is 1.

# Use this batching mechanism to spread the overhead of dispatching CICS over  
# several tasks. If the value is greater than 1 and CICS is in a system wait, CICS  
# is not posted for dispatch until the specified number of events has occurred.  
# Events include MRO requests from connected systems or DASD I/O and  
# CHANGE\_MODE processing. For these events, CICS is dispatched as soon as  
# one of the following occurs:

- # • The current batch fills up (the number of events equals MROBTCH)
- # • An ICV interval expires

# Therefore, ensure that the time interval you specify in the ICV parameter is low  
# enough to prevent undue delay to the system.

If CICS is dispatched for another reason, the current batch is dealt with in that dispatch of CICS.

**Note:** During periods of low utilization, a value of MROBTCH greater than 1 may result in increased transaction response times. Transactions issuing file I/O requests may be delayed due to increased FCIOWAIT. For further guidance information about the effect of MROBTCH on performance, see the *CICS Performance Guide*.

**MROFSE={NO|YES}**

specifies whether you want to extend the lifetime of the long-running mirror to keep it allocated until the end of the task rather than after a user syncpoint for function shipping applications.

**NO** The lifetime of the MRO long-running mirror is not extended.

**YES** The mirror task remains available to the application until the

end of the application's task. This extended long-running mirror saves the overhead of reattaching the mirror task following a user syncpoint.

This parameter is ignored for DPL requests (that is a DPL causes the session to be freed at the next syncpoint even if it has been kept for a previous sequence of syncpoints).

It should be used with caution, especially if DPL requests with SYNCONRETURN or TRANSID are used. For additional information, see the long running mirror sections of the *CICS Intercommunication Guide* and the *CICS Performance Guide*.

Do not specify this value in the front-end region when long running tasks might be used to function-ship requests. This is because a SEND session is unavailable for allocation to other tasks when unused. Specifying MROFSE=YES could prevent the connection from being released when contact has been lost with the back-end region, until the task terminates or issues a function-shipped request.

#  
#  
#  
#  
#  
#  
#  
#  
#  
#

**MROLRM={NO|YES}**

specifies whether you want to establish an MRO long-running mirror task.

**NO** The MRO long-running mirror task is not required.

**YES** The mirror transaction remains available to the application issuing the remote request. This long-running mirror saves the overhead of re-establishing communication with the mirror transaction if the application makes more function shipping requests in this unit of work.

For information about long-running mirror tasks, see the *CICS Intercommunication Guide*.

**MSGCASE={MIXED|UPPER}**

CICS messages handled by the CICS message domain are in mixed case. Specify this parameter to indicate how you want the message domain to display these mixed case messages.

**MIXED**

This is the default in the SIT; all messages displayed by the CICS message domain remain in mixed case.

**UPPER**

The message domain displays all mixed case messages in uppercase only.

**Note:** Mixed case output is not displayed correctly on Katakana display terminals and printers. Uppercase English characters appear correctly as uppercase English characters, but lowercase appears as Katakana symbols. If you have any Katakana terminals connected to your CICS region, specify MSGCASE=UPPER.

**MSGLVL={1|0}**

specifies the message level that controls the generation of messages to the console and JES message log.

**1** All messages are printed or displayed.

**0** Only critical errors or interactive messages are printed or displayed.

**MXT={5|number}**

specifies the maximum number, in the range 1 through 999, of **user** tasks CICS

allows to exist at any time. CICS queues requests for tasks above this number but does not action (attach) them until the number of tasks attached drops below the MXT limit.

Note that each active IIOF session requires two tasks.

You should review the region size specified on the REGION parameter for CICS address spaces. The increase in CICS use of virtual storage above the 16MB boundary means that you will probably need to increase the REGION parameter.

The introduction of the transaction isolation facility increases the allocation of some virtual storage above the 16MB boundary for CICS regions that are running with transaction isolation active.

If you are running with transaction isolation active, CICS allocates storage for task-lifetime storage in multiples of 1MB for user-key tasks that run above the 16MB boundary. (1MB is the minimum unit of storage allocation above the line for the EUDSA when transaction isolation is active.) However, although storage is allocated in multiples of 1MB above the 16MB boundary, MVS paging activity affects only the storage that is actually used (referenced), and unused parts of the 1MB allocation are not paged.

If you are running without transaction isolation, CICS allocates user-key task-lifetime storage above 16MB in multiples of 64KB.

The subspace group facility uses more real storage, as MVS creates for each subspace a page and segment table from real storage. The CICS requirement for real storage varies according to the transaction load at any one time. As a guideline, each task in the system requires 9KB of real storage, and this should be multiplied by the number of concurrent tasks that can be in the system at any one time (governed by the MXT system initialization parameter).

However, automatic DSA sizing removes the need for accurate storage estimates, with CICS dynamically changing the size of DSAs as demand requires.

**Note:** The MXT value does **not** include CICS system tasks.

**NATLANG=(E,x,y,z,...)**

specifies the single-character codes for the languages to be supported in this CICS run, selected from the codes in Table 22 on page 217.

**E** English, which is the *system* default (that is, is provided even if you do not specifically code E).

**x,y,z,...**

Specify the appropriate letters for the other supported languages that you require.

For the codes that you specify on this parameter, you must ensure that a DFHMET1x module (where x is the language code) is in a library in the STEPLIB DD concatenation of the CICS startup JCL. (For full language support, you must also provide other DFHMEyyx modules.) For information about using the message editing utility to create your own DFHMEyyx modules, see the *CICS Operations and Utilities Guide*.

English language support is provided, even if you do not specifically code E for English.

The first language code specifies the default language for those elements of CICS enabled to receive National Language Support (NLS) messages, such as

some destinations used for CICS messages, and the terminals or users not signed-on with an NLS code. The other language codes are provided to specify the language to be used for messages sent to terminals that are defined with the appropriate language support code. For example, coding NATLANG=(F,G,S) has the same effect as coding NATLANG=(F,G,E,S); that is, in both cases the default NLS language is French (F), and the languages English, German (G), and Spanish (S) are supported. (For such support, you would have to create and install the modules DFHMET1F, DFHMET1G, and DFHMET1S into a library in the STEPLIB DD concatenation of the CICS startup JCL.)

NLS is not available to CICS console messages, which continue to be in English only.

Table 22. Languages and codes supported by CICS

NATLANG code	NLS code	Language
A	ENG	Alternative English
Q	ARA	Arabic
1	BEL	Byelorussian
L	BGR	Bulgarian
B	PTB	Brazilian Portuguese
T DBCS	CHT	Traditional Chinese
C DBCS	CHS	Simplified Chinese
2	CSY	Czech
D	DAN	Danish
E	ENU	English
G	DEU	German
O	ELL	Greek
S	ESP	Spanish
W	FIN	Finnish
F	FRA	French
X	HEB	Hebrew
3	HRV	Croatian
4	HUN	Hungarian
J	ISL	Icelandic
I	ITA	Italian
K DBCS	JPN	Japanese
H DBCS	KOR	Korean
M	MKD	Macedonian
9	NLD	Dutch
N	NOR	Norwegian
5	PLK	Polish
P	PTG	Portuguese
6	ROM	Romanian
R	RUS	Russian
Y	SHC	Serbo-Croatian (Cyrillic)
7	SHL	Serbo-Croatian (Latin)
V	SVE	Swedish
Z	THA	Thai
8	TRK	Turkish
U	UKR	Ukrainian

Table 22. Languages and codes supported by CICS (continued)

NATLANG code	NLS code	Language
<b>Note:</b>		
DBCS denotes Double-Byte Character Set languages.		
The following language module suffixes are not supported by the message editing utility:		
<ul style="list-style-type: none"> <li>• E - English master data sets.</li> <li>• K - Japanese data sets, where translation is performed by IBM.</li> <li>• C - Simplified Chinese data sets, where translation is performed by IBM.</li> </ul>		
The NATLANG code is used as the suffix of the message modules for the associated language.		

**NCPLDFT**={DFHNC001 | name}

specifies the name of the default named counter pool to be used by the CICS region on calls it makes to a named counter server. If CICS cannot determine, from the named counter options table, the pool name required by an EXEC CICS named counter command, CICS uses the default name specified on the NCPLDFT parameter.

**Note:** This parameter is relevant to references to a named counter server made through the EXEC CICS API only. It not used by the named counter call interface.

**DFHNC001**

This is the default name that CICS uses as the named counter pool name if you omit the NCPLDFT system initialization parameter.

**name** Specifies the 8-character name to be used by CICS as the default pool name in connection with named counter API commands, when the name cannot be resolved by the named counter options table.

**NEWSIT**={YES | NO}

specifies whether CICS is to load the specified SIT, and enforce the use of all system initialization parameters, modified by any system initialization parameters provided by PARM, SYSIN, or the system console, even in a warm start. Enforcing the use of system initialization parameters in this way overrides any parameters that may have been stored in a warm keypoint at shutdown.

However, there are some exceptions. The following system initialization parameters are always ignored in a warm start, even if they are supplied by PARM, SYSIN, or the console:

- CSDACC
- CSDBUFND
- CSDBUFNI
- CSDDISP
- CSDDSN
- CSDFRLOG
- CSDINTEG
- CSDJID
- CSDLRNO
- CSDRECOV
- CSDRLS
- CSDSTRNO
- FCT
- GRPLIST

In a warm restart, CICS uses the **installed** resource definitions saved in the CICS global catalog at warm shutdown, and therefore the CSD, FCT, and GRPLIST parameters are ignored. (At CICS startup, you can only modify installed resource definitions, including file control table entries, or change to a new FCT, by performing a cold start of CICS with START=COLD.)

For more information about the use of the NEWSIT parameter, see “Controlling start and restart” on page 278.

### Restrictions

You can specify the NEWSIT parameter in PARM, SYSIN, or CONSOLE only.

**OFFSITE={NO|YES}**

specifies whether CICS is to restart in off-site recovery mode; that is, a restart is taking place at a remote site.

**Note:** For a successful off-site restart, the log records of the failed CICS region must be available at the remote site. CICS does not provide a facility for shipping log records to a remote backup site, but you can use a suitable vendor product to perform this function. See the relevant product documentation for other procedures you need to follow for a remote site restart.

See the *CICS Recovery and Restart Guide* for more information about remote site recovery.

**NO** CICS will not perform the special restart processing required for remote site recovery.

**YES** CICS will perform an off-site restart at a remote site following a disaster at the primary site. CICS performs this special processing for an off-site restart, because some information (for example, a VSAM lock structure) is not available at the remote site.

CICS performs an emergency restart, even if the global catalog indicates that CICS can do a warm start. OFFSITE=YES is valid with START=AUTO only, and CICS initialization is terminated if you specify START=COLD or INITIAL.

### Restrictions

You can specify the OFFSITE parameter in PARM, SYSIN, or CONSOLE only.

**OPERTIM={120|number}**

specifies the write-to-operator timeout value, in the range 0 through 86400 seconds (24 hours). This is the maximum time (in seconds) that CICS waits for a reply before returning control to this transaction. For information about using the write-to-operator timeout value, see the *CICS Application Programming Reference*.

**OPNDLIM={10|number}** (Not required for currently supported releases of VTAM.)

specifies the open destination and close destination request limit. This limit is used to restrict the number of concurrent OPNDSTs and CLSDSTs to prevent VTAM from running out of space in the CICS region. The limit may be any value in the range 0 through 999. When large values are used for OPNDLIM, the value on the EDSALIM system initialization parameter and the value on the MVS REGION parameter may need to be adjusted to ensure that enough operating system storage is available. For information about adjusting these parameters, see the *CICS Performance Guide*.

**PARMERR={INTERACT|IGNORE|ABEND}**

specifies what action you want to follow if CICS detects incorrect system initialization parameter overrides during initialization.

**Note:** When specified as an override, this parameter affects only subsequent system initialization parameter overrides. Errors in earlier system initialization parameter overrides are dealt with according to the PARMERR system initialization parameter value in the SIT.

**INTERACT**

Enables the operator to communicate with CICS through the console and correct parameter errors.

**Note:** INTERACT is overridden with IGNORE in the following cases:

- If errors are found in PARM or SYSIN for system initialization parameter overrides that are not allowed to be entered from the console
- In certain circumstances, in response to invalid data when you have been trying to correct a previous invalid system initialization parameter keyword or value

**IGNORE**

CICS ignores errors, and tries to complete initialization.

**ABEND**

CICS abends.

**PDI={30|decimal-value}**

specifies the XRF primary delay interval, in seconds, in a SIT for an active CICS region. The minimum delay that you can specify is 5 seconds. This is the time that must elapse between the (apparent) loss of the surveillance signal in the alternate CICS region, and any reaction by the active CICS region. The corresponding parameter for the alternate CICS region is ADI. PDI and ADI need not have the same value.

**PDIR={NO|yes|xx}**

specifies a suffix for the PDIR list. A PDIR is a list of program specification blocks (PSBs) that define, for DL/I, the use of databases by application programs. This is applicable only if DL/I remote support is being used. (See also page “Defining CICS resource table and module keywords” on page 162.) Specifying a value other than NO implies to CICS that remote DLI support is required.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

**PGAICTLG={MODIFY|NONE|ALL}**

specifies whether autoinstalled program definitions should be cataloged. While CICS is running, you can set whether autoinstalled programs should be cataloged dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

**MODIFY**

Autoinstalled program definitions are cataloged only if the program definition is modified by a SET PROGRAM command subsequent to the autoinstall.

**NONE** Autoinstalled program definitions are not cataloged. This gives a faster CICS restart (warm and emergency) compared with the MODIFY or ALL

options, because CICS does not reinstall definitions from the global catalog. Definitions are autoinstalled on first reference.

**ALL** Autoinstalled program definitions are written to the global catalog at the time of the autoinstall, and following any subsequent modification.

**PGAEXIT={DFHPGADX|name}**

specifies the name of the program autoinstall exit program. While CICS is running, you can set the name of the program autoinstall exit program dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

**PGAIPGM={INACTIVE|ACTIVE}**

specifies the state of the program autoinstall function at initialization. While CICS is running, you can set the status of program autoinstall dynamically, by using either the EXEC CICS SET SYSTEM or CEMT SET SYSTEM command.

**INACTIVE**

The program autoinstall function is disabled.

**ACTIVE**

The program autoinstall function is enabled.

**PGCHAIN=character(s)**

specifies the character string that is identified by terminal control as a BMS terminal page-chaining command. It can be 1 through 7 characters. For more information about the character string, see the notes on page 1.

**PGCOPY=character(s)**

specifies the character string that is identified by terminal control as a BMS command to copy output from one terminal to another. It can be 1 through 7 characters. For more information about the character string, see the notes on page 1.

**PGPURGE=character(s)**

specifies the character string that is identified by terminal control as a BMS terminal page-purge command. It can be 1 through 7 characters. For more information about the character string, see the notes on page 1.

**PGRET=character(s)**

specifies the character string that is recognized by terminal control as a BMS terminal page-retrieval command. It can be 1 through 7 characters.

**Notes:**

1. Each character string is unique with respect to the leading characters of every other transaction identification defined in the CSD. A command requested by a single character precludes the use of all other transaction identifications starting with this character.
2. In pseudoconversational mode, each character string is unique with respect to the leading characters of any terminal input message.
3. A field-separator or other suitable delimiter may be specified in each character string to separate this command code from the remainder of the paging command when entered by an operator. For example:

```
PGCHAIN = X/  
PGCOPY = C/  
PGPURGE = T/  
PGRET = P/
```

This reduces the risk of creating a nonunique command. (See Note 1.)

**Restrictions**

If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET in the SIT, the characters you choose must not include any of the following: ( ) ' =

If you specify PGCHAIN, PGCOPY, PGPURGE, or PGRET as a PARM, SYSIN, or console parameter, do not enclose the characters in quotation marks. The characters you choose must not include an embedded blank or any of the following: ( ) ' =

4. PGCHAIN, PGCOPY, PGPURGE, and PGRET are required only if full function BMS is being used. For information about the BMS page retrieval transaction CSPG, see *CICS Supplied Transactions*.
5. CICS always processes a paging command entered by the operator before initiating a transaction invoked by an EXEC CICS RETURN command with the TRANSID option.

**PLTPI={NO|xx|YES}**

specifies a program list table, which contains a list of programs to be executed in the final stages of system initialization (see page “Defining CICS resource table and module keywords” on page 162). You can use the system initialization parameter INITPARM to pass parameters to those programs.

For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

**PLTPISEC={NONE|CMDSEC|RESSEC|ALL}**

specifies whether or not you want CICS to perform command security or resource security checking for PLT programs during CICS initialization. The PLT programs run under the authority of the userid specified on PLTPIUSR, which must be authorized to the appropriate resources defined by PLTPISEC.

**NONE** You do not want any security checking on PLT initialization programs.

**CMDSEC**

You want CICS to perform command security checking only.

**RESSEC**

You want CICS to perform resource security checking only.

**ALL** You want CICS to perform both command and resource security checking.

**Restrictions** You can specify the PLTPISEC parameter in the SIT, PARM, or SYSIN only.

**PLTPIUSR=userid**

specifies the userid that CICS is to use for security checking for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs, as defined by the PLTPISEC parameter.

PLT programs are run under the CICS internal transaction, CPLT. Before the CPLT transaction is attached, CICS performs a surrogate user check against the CICS region userid (the userid under which the CICS region is executing). This is to ensure that the CICS region is authorized as a surrogate for the userid specified on the PLTPIUSR parameter. This ensures that you cannot arbitrarily specify any PLT userid in any CICS region—each PLT userid must first be authorized to the appropriate CICS region.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must be authorized to all the resources referenced by the PLT programs.

**Restrictions** You can specify the PLTPIUSR parameter in the SIT, PARM, or SYSIN only.

**PLTSD={NO|xx|YES}**

specifies a program list table that contains a list of programs to be executed during system termination (see page “Defining CICS resource table and module keywords” on page 162).

**PRGDLAY={0|hhmm}**

specifies the BMS purge delay time interval that is added to the specified delivery time to determine when a message is to be considered undeliverable and therefore purged. This time interval is specified in the form “hhmm” (where “hh” represents hours from 00 to 99 and “mm” represents minutes from 00 to 59). If PRGDLAY is not coded, or is given a zero value, a message remains eligible for delivery either until it is purged or until temporary storage is cold started.

**Note:** If you specify PRGDLAY as a SIT override, you must still specify a 4-character value (for example 0000).

The PRGDLAY facility requires the use of full function BMS. Note also that you must code a PRGDLAY value if you want the ERRTERMIERRTERM(name) parameter on EXEC CICS ROUTE commands to be operative. For programming information about notification of undelivered messages, see the *CICS Application Programming Reference*.

The PRGDLAY value determines the interval between terminal page clean-up operations. A very low value causes the CSPQ transaction to be initiated continuously, and can have a detrimental effect on task-related resources. A zero value stops CSPQ initiating terminal page clean-up. However, this can cause messages to stay in the system forever, resulting in performance problems with long AID queues or lack of temporary storage. The actual purge delay time interval specified is dependent on individual system requirements.

**PRINT={NO|YES|PA1|PA2|PA3}**

specifies the method of requesting printout of the contents of a 3270 screen.

**NO** Screen copying is not required.

**YES** Screen copying can be requested by terminal control print requests only.

**PA1, PA2, or PA3**

Screen copying can be requested by terminal control print request, or by using the PA (program attention) key specified.

The PA key specified by this parameter must not be specified by the TASKREQ option of the RDO TRANSACTION definition or be used for 3270 single keystroke retrieval.

When YES, PA1, PA2, or PA3 is specified, transaction CSPP is initiated which invokes program DFHP3270. The transaction and programs are defined in the CSD group DFHHARDC. In the case of 3270 and LUTYPE2 logical units, the resources defined in CSD group DFHVAMP are required.

The 3270 print-request facility allows either the application program or the terminal operator to request a printout of data currently displayed on the 3270 display.

If CSPP is invoked to print the screen contents at an associated VTAM printer, the screen size of the printer is chosen according to the screen size defined in the profile for the transaction CSPP. The CICS-supplied definitions use the

default screen size. Therefore, if you want DFHP3270 to use the alternate screen size of the printer, you must alter the screen size defined in the profile for the transaction CSPP. For information about defining profiles for transactions, see *CICS Supplied Transactions*.

For a VTAM 3270 display without the printer-adapter feature, the PRINT request prints the contents of the display on the first available 3270 printer specified by PRINTER and ALTPRINTER options of the RDO TERMINAL definition. For a printer to be considered available, it must be in service and not currently attached to a task. It is not necessary for the printer to be on the same control unit.

In an MRO environment, the printer must be owned by the same system as the VTAM 3270 display.

For the 3275 with the printer-adapter feature, the PRINT request prints the data currently in the 3275 display buffer on the 3284 Model 3 printer attached to the 3275.

The format of the print operation depends on the size of the display buffer. For a 40-character wide display, the print format is a 40-byte line, and for an 80-character wide display the format is an 80-byte line.

For the 3270 compatibility mode logical unit of the 3790 (if the logical unit has the printer-adapter feature specified), the PRINT request prints the contents of the display on the first printer available to the 3790. The allocation of the printer to be used is under the control of the 3790.

For 3274, 3276, and LUTYPE2 logical units with the printer-adapter feature, the PRINT request prints the contents of the display on the first printer available to the 3270 control unit. The printer to be allocated depends on the printer authorization matrix.

For the 3270 compatibility mode logical unit without the printer-adapter feature, see the preceding paragraph on VTAM 3270 displays without the printer-adapter feature.

**PRTYAGE={32768 | value}**

specifies the number of milliseconds to be used in the priority aging algorithm for incrementing the priority of a task. The value can be in the range 0 through 65535, and 32768 is the default.

The priority aging factor is used to increase the effective priority of a task according to the amount of time it is held on a ready queue. The value represents the number of milliseconds that must elapse before the priority of a waiting task can be adjusted upwards by 1. For example, if you code PRTYAGE=3000, a task has its priority raised by 1 for every 3000 milliseconds it is held on the ready queue. Thus a high value for PRTYAGE results in a task being promoted very slowly up the priority increment range, and a low value enables a task to have its priority incremented quickly.

If you specify a value of 0, the priority aging algorithm is not used (task priorities are not modified by age) and tasks on the ready queue are handled according to the user assigned priority.

**PRVMOD={name | (name, name . . . name)}**

specifies the names of those modules that are not to be used from the LPA.

The operand is a list of 1-to 8-character module names. This enables you to use a private version of a CICS nucleus module in the CICS address space, and not a version that might be in the LPA. For information about PRVMOD, see the *CICS Transaction Server for z/OS Installation Guide*.

**Restrictions** You can specify the PRVMOD parameter in PARM, SYSIN, or CONSOLE only.

**PSBCHK={NO|YES}**

specifies whether CICS is to perform PSB authorization checks for remote terminal users who use transaction routing to initiate a transaction in this CICS region (to access an attached IMS system).

**NO** The remote link is checked, but no check is made against the remote terminal. This is the default.

**YES** The remote link is checked, and the remote terminal is also checked if RESSEC(YES) is coded in the definition of the transaction in the CSD.

**Restrictions** You can specify the PSBCHK parameter in the SIT, PARM, or SYSIN only.

**Note:** If you require DL/I security checking, you must specify the XPSB system initialization parameter as XPSB=YES or XSPB=name. For further information about the XPSB system initialization parameter, see “XPSB ” on page 261.

**PSDINT={0|hhmmss}**

specifies the persistent session delay interval. This delay interval specifies if, and for how long, VTAM is to hold sessions in a recovery-pending state if CICS fails. The value for hours can be in the range 0 through 23; the minutes and seconds in the range 00 through 59 inclusive.

This value can be overridden during CICS execution (and hence change the action taken by VTAM if CICS fails).

**0** If CICS fails, sessions are terminated. This is the default.

**hhmmss**

A persistent session delay interval from 1 second up to the maximum of 23 hours 59 minutes and 59 seconds. If CICS fails, VTAM holds sessions in recovery pending state for up to the interval specified on the PSDINT system initialization parameter.

Specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

The interval you specify must cover the time from when CICS fails to when the VTAM ACB is opened by CICS during the subsequent emergency restart.

VTAM holds all sessions in recovery pending state for up to the interval specified (unless they are unbound through path failure or VTAM operator action, or other-system action in the case of intelligent LUs). The PSDINT value used must take account of the types and numbers of sessions involved.

You must exercise care when specifying large PSDINT values because of the problems they may give in some environments, in particular:

- Dial-up sessions—real costs may be incurred
- LU6.2 sessions to other host systems—such systems may become stressed

**Notes:**

1. When specifying a PSDINT value, you must consider the number and, more particularly, the nature of the sessions involved. If LU6.2 sessions to other host systems are retained in recovery pending state, the other host systems

may experience excessive queuing delays. This point applies to LU6.1 sessions which are retained until restart (when they are unbound).

2. The PSDINT parameter is incompatible with the XRF=YES parameter. If XRF=YES is specified, the PSDINT parameter is ignored.

**PSTYPE={SNPS|MNPS}**

specifies whether CICS is running with VTAM single node persistent sessions (SNPS) or multi node persistent sessions (MNPS). Code this parameter if you are using VTAM MNPS and you wish to recover sessions when the VTAM ACB is opened after a VTAM failure. You should read the VTAM Network Implementation Guide to see how VTAM should be set up to use MNPS and under what conditions sessions persist for MNPS.

**PVDELAY={30|number}**

specifies the persistent verification delay as a value in the range 0 through 10080 minutes (up to 7 days). PVDELAY defines how long entries can remain in the signed-on-from lists for those connections for which persistent verification is specified in a connection resource definition. If you specify PVDELAY=0, entries are deleted immediately after use.

For information about the use of PVDELAY, see the *CICS Performance Guide*.

**QUIESTIM={240|number}**

specifies a timeout value for data set quiesce requests.

In a busy CICSplex, it is possible for the default timeout to expire before the quiesce request has been processed by all the CICS regions, even though there is nothing wrong. If the quiesce operation is not completed when the timeout period expires, SMS VSAM cancels the quiesce. If you find that timeout is occurring too frequently, increase the timeout value.

Specify the timeout value as a number of seconds. The default value is 240 seconds (4 minutes)

The maximum timeout value you can specify is 3600 (1 hour).

**RAMAX={256|number}**

specifies the size in bytes of the I/O area allocated for each RECEIVE ANY issued by CICS, in the range 0 through 32767 bytes.

**Note:** If you are using APPC, do not code a value less than 256; otherwise, the results are unpredictable.

For information about coding this parameter, see the *CICS Performance Guide*.

**RAP00L={50|value1|(value1,value2,FORCE)}**

specifies the size of the CICS receive any pool. value1 is the number of fixed request parameter lists (RPLs), receive any control elements (RACEs), and receive any input areas (RAIAs) that are to be generated whether or not CICS uses the high performance option (HPO). value1, in the range 1 through 999, is also the number that are active in a non-HPO system; value2, in the range 0 through 999, is the number that are active in an HPO system. The default for value1 in the DFHSIT macro is 2. The default for value2 is calculated from value1 as follows:

- If value1 = 1, value2 = 1
- If value1 ≤ 5, value2 = (value1 minus 1)
- If value1 ≥ 6 and ≤ 49, value2 = 5
- If value1 ≥ 50, value2 is 10 per cent of value1

**Note:** You should code value1 equal to or greater than value2; if you code value1 less than value2, CICS forces value2 equal to value1.

If you omit the RAPOOL parameter altogether, RAPOOL=(50,1) is assumed. CICS maintains n VTAM RECEIVE ANYs, where n is either the RAPOOL “number active” value, or the MXT value minus the number of active tasks, whichever is the smaller. For example, in a non-HPO system:

If RAPOOL=2, MXT=50, active tasks = 45 then RECEIVE ANY = 2  
If RAPOOL=10, MXT=50, active tasks = 45 then RECEIVE ANY = 5  
If RAPOOL=10, MXT=50, active tasks = 35 then RECEIVE ANY = 10

or in an HPO system:

If RAPOOL=(20,10), MXT=50, active tasks = 45 then RECEIVE ANY = 5

FORCE tells CICS to free up Receive\_Any\_RPLs if they are stalled. CICS decides that the Receive\_Any\_RPLs are stalled if all the RA RPLs have been posted but the TCTTE for each one is waiting for a response from a VTAM terminal or session for 10 dispatches of the TCP (CSTP) task.

This typically happens only if a protocol error has occurred, and sessions are waiting for a response; for example, to a BID SHUTD request from CICS.

Each session is unbound, the Receive\_Any data is lost and the RA RPL is reissued thus allowing VTAM activity to continue: Message DFHZC4949 is issued for each session affected.

Consider increasing the size of the RAPOOL before resorting to the use of FORCE.

If FORCE is not specified and a Receive\_Any stall occurs, DFHZC2118 is written to the console for each session affected.

If FORCE is specified in the SIT, and RAPOOL is supplied as an override, you must again specify FORCE as otherwise it defaults to FORCE not specified.

The number of RECEIVE ANYs needed depends on the expected activity of the system, the average transaction lifetime, and the MAXTASK value specified. For information about coding this parameter, see the *CICS Performance Guide*.

**RDSASZE={0K|number}**

specifies the size of the RDSA. The default size is 0, indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

specify number as an amount of storage in the range 0 to 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the RDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**RENTPGM={PROTECT|NOPROTECT}**

specifies whether you want CICS to allocate the read-only DSAs, RDSA and ERDSA, from read-only key-0 protected storage. The permitted values are PROTECT (the default), or NOPROTECT:

## **PROTECT**

CICS obtains the storage for the read-only DSAs from key-0 protected storage.

## **NOPROTECT**

CICS obtains the storage from CICS-key storage, effectively creating two more CICS DSAs (CDSA and ECDSA). This allows programs eligible for the read-only DSAs to be modified by programs that execute in CICS key.

You are recommended to specify `RENTPGM=NOPROTECT` for development regions only, and to specify `RENTPGM=PROTECT` for production CICS regions.

## **RESP={FME|RRN}**

specifies the type of request that CICS terminal control receives from logical units.

**FME** Function management end is the default.

**RRN** Reached recovery node.

## **RESSEC={ASIS|ALWAYS}**

specifies whether you want CICS to honor the RESSEC option specified on a transaction's resource definition.

**ASIS** CICS honors the RESSEC option defined in a transaction's resource definition. CICS calls its resource security checking routine only when RESSEC(YES) is specified in a transaction resource definition. This is normally a sufficient level of control, because often you will need only to control the ability to execute a transaction.

## **ALWAYS**

CICS overrides the RESSEC option, and always calls its resource security checking routine to issue the appropriate call to the SAF interface.

Use this option only if you need to control or audit all accesses to CICS resources. Using this option can significantly degrade performance.

**Restrictions** You can specify the RESSEC parameter in the SIT, PARM, or SYSIN only.

## **RLS={NO|YES}**

specifies whether CICS is to support VSAM record-level sharing (RLS).

**NO** RLS support is not required in this CICS region. Files whose definitions specify RLSACCESS(YES) will fail to open, with an error indicating that RLS access is not supported. You should not specify RLS=NO if you have files that you want to open in RLS access mode (including the CSD).

**YES** RLS support is required in this CICS region. During initialization, CICS automatically registers with an SMSVSAM control ACB to enable RLS access to files opened with RLSACCESS(YES).

## **RLSTOLSR={NO|YES}**

specifies whether CICS is to include files that are to be opened in RLS mode when calculating the number of buffers, strings, and other resources for an LSR pool. CICS performs this calculation only when you have not explicitly defined an LSRPOOL resource definition that corresponds to an LSRPOOLID in a file definition. CICS calculates and builds a default LSR pool only when it is opening the first file in LSR mode that references the default pool.

**NO** CICS is not to include files opened in RLS mode, and which also specify an LSRPOOLID, when it is building default LSR pools. Files defined with RLSACCESS(YES) are ignored when CICS is scanning file entries looking for files that specify an LSR pool it is about to build using default values.

If the LSR pools referenced by LSRPOOLIDs in your file resource definitions are defined explicitly by LSRPOOL resource definitions, you should specify RLSTOLSR=NO.

**YES** CICS is to include in its calculation, when building default LSR pools, files that specify both RLSACCESS(YES) and an LSRPOOLID.

Note that an LSR pool built including files that are opened in RLS mode is larger than necessary initially. This option is provided to ensure that, if files are subsequently switched to LSR, the LSR pool is adequate for the extra files. You should specify RLSTOLSR=YES only if both of the following conditions are true:

1. You do not define LSR pools explicitly, relying instead on CICS obtaining a default set of values for you.
2. You have files that are sometimes accessed in RLS mode and sometimes accessed in non-RLS mode (although this is generally not recommended).

The RLSTOLSR parameter is provided to support files that are normally opened in RLS mode, but which may be closed and then switched to LSR mode.

If LSR pools are not defined explicitly using LSRPOOL resource definitions, CICS calculates the resources needed for an LSR pool using default attributes. CICS performs this calculation when opening the first file that specifies an LSR pool that is not explicitly defined. To calculate a default LSR pool, CICS scans all the file entries to count all the files that specify the same LSRPOOLID. The size of an LSR pool built dynamically in this way remains fixed until all files that reference the LSR pool are closed. After all files have been closed, another request to open a file with the same LSRPOOLID causes CICS to recalculate the size.

If you add files to the system *after* the LSR calculation has been performed there may be insufficient storage available to enable CICS to open a file that specifies a default pool. This situation could occur if files are opened initially in RLS mode and later closed and reopened in LSR mode. There are two ways to ensure that enough resources are built into the LSR pool to support subsequent switches of files from RLS to LSR:

1. You can explicitly define LSRPOOL resource definitions that correspond to the LSRPOOLIDs on file definitions, removing the need for CICS to calculate default values.
2. You can specify RLSTOLSR=YES to force CICS to include RLS files when calculating defaults.

**RMTRAN=({CSGM|name1}[,{CSGM |name2}])**

specifies the name of the transaction that you want an alternate CICS to initiate when logged-on class 1 terminals, which are defined with the attribute RECOVNOTIFY(TRANSACTION) specified, are switched following a takeover. This parameter is applicable only on an alternate CICS region.

If you do not specify a name here, CICS uses the CSGM transaction, the default CICS good morning transaction.

**name1**

This is the transaction that CICS initiates at terminals that do **not** remain signed-on after the takeover (that is, they are still connected to CICS, but are signed off).

**name2**

This is the transaction that CICS initiates at terminals that remain signed-on after the takeover. If you specify only name1, CICS uses the CSGM transaction as the default for name2.

#  
#

If you are using VSAM persistent sessions, the name2 transaction is ignored and the name1 transaction is always initiated.

**RRMS=NO|YES**

specifies whether CICS is to register as a resource manager with recoverable resource management services (RRMS).

**NO** You do not require RRMS support.

**YES** You require RRMS support to enable DPL requests to be coordinated by resource recovery services (RRS).

**Note:** If you specify RRMS=YES, ensure that the DFHRXSVC module is available during CICS initialization. This module, which provides RRMS authorized services, is supplied in the SDFHLINK library. For information about this link list library, see the *CICS Transaction Server for z/OS Installation Guide*.

**RST={NO|xx|YES}**

specifies a recoverable service table suffix. (See page “Defining CICS resource table and module keywords” on page 162.) For information about coding the macros for this table, see the *CICS Resource Definition Guide*.

If you are running CICS with XRF=YES, and you are using DBCTL, you must specify an RST if you want XRF support for DBCTL. For information about the use of the RST in a CICS-DBCTL environment with XRF=YES, see the *CICS IMS Database Control Guide*.

**RSTSIGNOFF={NOFORCE|FORCE}**

specifies whether all users signed-on to the active CICS region are to remain signed-on following a persistent sessions restart or an XRF takeover. It applies to the following events:

- A persistent sessions restart, where PSDINT=*value* and PSTYPE=SNPS or MNPS are specified, and the restart follows a CICS abnormal or immediate shutdown.
- A persistent sessions restart, where PSDINT=*value* and PSTYPE=MNPS are specified, and terminal sessions are recovered as a result of a VTAM restart.
- An XRF takeover, where XRF=YES is specified.

**NOFORCE**

Do not sign off users, unless FORCE is specified on either:

- The RSTSIGNOFF parameter in the TYPETERM definition referenced by the user's terminal definition.
- The XRFSOFF parameter in the CICS segment of the user's RACF profile.

Thus for a user to remain signed on after a persistent sessions restart or an XRF takeover, NOFORCE must be specified as a system initialization parameter, on the TYPETERM definition, and in the CICS segment.

## FORCE

Sign off all users regardless of the options specified on:

- The RSTSIGNOFF parameter in the TYPETERM definition referenced by the user's terminal definition.
- The XRFSOFF parameter in the CICS segment of the user's RACF profile.

See the *CICS RACF Security Guide* for information about user profile options in the CICS segment, and see the *CICS Resource Definition Guide* for information about the TYPETERM resource definition.

## RSTSIGTIME={5|decimal-value}

Specifies the time-out delay interval for signon retention during a persistent sessions restart or an XRF takeover. You can specify a 1-to-6 digit time in hours, minutes and seconds, up to the maximum time of 23 hours 59 minutes 59 seconds. If you specify less than six digits, CICS pads the value with leading zeros. Thus a value of 500 is taken as five minutes exactly.

RSTSIGTIME is counted from the time when CICS failed. Note that the time of failure cannot be determined with complete accuracy.

If you specify NOFORCE on all the appropriate parameters to enable a user to remain signed on, but the persistent sessions restart or XRF takeover takes longer than the specified on the RSTSIGTIME parameter, CICS ensures users do not remain signed on after the delay period expires.

**500** Five minutes is the default value.

*time* This is the time, in the range 0 through 23 hours 59 minutes 59 seconds, during which CICS permits users to remain signed on during a persistent sessions restart or an XRF takeover. The period is measured as follows:

- For a persistent sessions restart, the period is the time from the CICS failure and the time when the user starts working on the terminal. If the specified time expires before the user starts working on the terminal, users signed on at the time CICS failed are not signed on again after restart.
- For an XRF takeover, the period is the time from when the takeover is initiated to the time at which the alternate CICS has completed takeover and is ready to process user transactions. If the takeover takes longer than the specified period, all users signed on at the time the takeover was initiated are signed off.

A value of 0 means there is no time-out delay, and terminals are not signed on after a persistent sessions restart or XRF takeover, which means that RSTSIGTIME=0 has the same effect as coding RSTSIGNOFF=FORCE.

When XRF is in use with non-XRF-capable terminals, take into account any AUTCONN delay period when setting the value for RSTSIGTIME. For example, you may need to increase the time specified on RSTSIGTIME to allow for the delay up to the start of the CXRE transaction imposed by the AUTCONN parameter; otherwise, terminals could be signed off too early.

## RUWAPool={NO|YES}

specifies the option for allocating a storage pool the first time a program invoked by Language Environment runs in a task.

**NO** CICS disables the option and provides no RUWA storage pool. Every

EXEC CICS LINK to a program that runs under Language Environment results in a GETMAIN for RUWA storage.

**YES** CICS creates a pool of storage the first time a program invoked by Language Environment runs in a task. This provides an available storage pool that reduces the need to GETMAIN and FREEMAIN run-unit work areas (RUWAs) for every EXEC CICS LINK request.

**Note:** This applies only to application programs running with the Language Environment run-time option ALL31(ON). RUWAPool=YES has no effect on application programs running with the Language Environment run-time option ALL31(OFF).

**SDSASZE={0K|number}**

specifies the size of the SDSA. The default size is 0, indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

specify number as an amount of storage in the range 0 to 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB, CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the SDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**SDTRAN={CESD|name\_of\_shutdown\_tran|NO}**

specifies the name of the shutdown transaction to be started at the beginning of normal and immediate shutdown.

The shutdown transaction enables CICS to shut down in a controlled manner, within a reasonable period of time. For example, you can use it to purge and backout long-running tasks, while ensuring that as many tasks as possible commit or backout cleanly. For information about the CICS-supplied program, DFHCESD, started by the default shutdown transaction, CESD, and how to use it as the basis for your own transaction, see the *CICS Operations and Utilities Guide*.

**Notes:**

1. The transaction runs under the userid authority of the issuer of the shutdown command.
2. If the program named by the shutdown transaction cannot be loaded, CICS waits indefinitely for all user tasks to complete. *This happens on an immediate, as well as on a normal, shutdown.*

**CESD** Starts the CICS-supplied program DFHCESD.

**name\_of\_shutdown\_transaction**

The 1-to 4-character name of your own shutdown transaction.

**NO** No shutdown transaction is to be run. On a normal shutdown, CICS waits indefinitely for all user tasks to complete.

**SEC={YES|NO}**

specifies what level of external security you want CICS to use.

**YES** You want to use full external security. CICS requires the appropriate

level of authorization for the access intent: a minimum of READ permission for read intent, and a minimum of UPDATE permission for update intent.

**Note:** You must also ensure that the default userid (CICSUSER or another userid specified on the DFLTUSER system initialization parameter) has been defined to RACF.

If command security checking is defined for CICS SP-type commands, then specifying SEC=YES means that the appropriate level of authority is checked for; therefore:

- A check for READ authority is made for INQUIRE and COLLECT commands.
- A check for UPDATE authority is made for SET, PERFORM, and DISCARD commands.

For the results of the interaction between the access intent of the user application, and the permission defined to RACF, see Table 23.

**NO** You do not want CICS to use an external security manager. All users have access to all resources, whether determined by attempts to use them or by the QUERY SECURITY command. Users are not allowed to sign on or off.

**Note:** With MRO bind-time security, even if you specify SEC=NO, the CICS region userid is still sent to the secondary CICS region, and bind-time checking is still carried out in the secondary CICS region. For information about MRO bind-time security, see the *CICS RACF Security Guide*.

Define whether to use RACF for resource level checking by using the XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, and XTST system initialization parameters. Define whether to use RACF for transaction-attach security checking by using the XTRAN system initialization parameter. Define whether to use RACF for enterprise bean method authorization checks by using the XEJB system initialization parameter. Define whether RACF session security can be used when establishing APPC sessions by using the XAPPC system initialization parameter.

For information on defining command security checking for CICS SP-type commands, and about CICS security in general, see the *CICS RACF Security Guide*.

For programming information about the use of external security for CICS system commands, see *CICS System Programming Reference*.

Table 23. Results of RACF authorization requests (with SEC=YES)

Access Permission defined to RACF for CICS user	Access intent in application	
	READ	UPDATE
NONE	Refused	Refused
READ	Permitted	Refused
UPDATE	Permitted	Permitted

**Restrictions** You can specify the SEC parameter in the SIT, PARM, or SYSIN only.

**Note:** If you are using preset terminal security (see *CICS RACF Security Guide*), and you perform a warm start with SEC=NO and then again with SEC=YES, you must reinstall the terminal definition to preserve the preset user ID that is replaced by the default user ID when security is switched off.

**SECPRFX={NO|YES|prefix}**

specifies whether CICS is to prefix the resource names in any authorization requests to the external security manager.

**NO** CICS does not use prefixes on any resource names.

**YES** CICS prefixes all resource names with the CICS region user ID. This is the user ID under which the CICS job runs. It is one of the following:

- If CICS is a batch job, it is the user ID corresponding to the USER parameter of the CICS JOB statement.
- If CICS is a started task, it is the user ID associated with the name of the started procedure in the RACF ICHRIN03 table.
- If CICS is a started job, it is the user ID specified in the user parameter of the STDATA segment of a STARTED general resource class profile.

For more information, see the *CICS RACF Security Guide*.

**prefix** CICS prefixes all resource names with the string you specify. It can be any string of 1 to 8 upper case alphanumeric characters except NO or YES, and must start with an alphabetic character.

**Restrictions** You can specify the SECPRFX parameter in the SIT, PARM, or SYSIN only.

The SECPRFX parameter is effective only if you specify YES for the SEC system initialization parameter.

**SIT=xx**

specifies the suffix, if any, of the system initialization table that you want CICS to load at the start of initialization. If you omit this parameter, CICS loads the unsuffixed table, DFHSIT, which is pregenerated with all the default values. This default SIT (shown in “The default system initialization table” on page 263) is in CICSTS31.CICS.SDFHAUTH, and its source, named DFHSIT\$\$, is in CICSTS31.CICS.SDFHSAMP.

**Restrictions** You can specify the system initialization parameter anywhere in PARM or SYSIN, or as the **first** parameter entry at the CONSOLE.

**SKRxxxx='page-retrieval-command'**

specifies that a single-keystroke-retrieval operation is required. 'xxxx' specifies a key on the 3270 keyboard which, during a page retrieval session, is to be used to represent a page retrieval command. The valid keys are PA1 through PA3, and PF1 through PF36. Thus up to 39 keys can be specified in this way (each by a separate command).

The 'page-retrieval-command' value represents any valid page retrieval command, and must be enclosed in apostrophes. It is concatenated to the character string coded in the PGRET parameter. The combined length must not exceed 16 characters.

**Note:** If full function BMS is used, all PA keys and PF keys are interpreted for page retrieval commands, even if some of these keys are not defined.

**SNSCOPE={NONE|CICS|MVSIMAGE|SYSPLEX}**

specifies whether a userid can be signed on to CICS more than once, within the scope of:

- A single CICS region
- A single MVS image
- A sysplex

The signon SCOPE is enforced with the MVS ENQ macro where there is a limit on the number of outstanding MVS ENQs per address space. If this limit is exceeded, the MVS ENQ is rejected and CICS is unable to detect if the user is already signed on. When this happens, the signon request is rejected with message DFHCE3587. See the *OS/390 MVS Programming: Authorized Assembler Services Guide* for guidance on increasing the MVS ENQ limit.

**NONE** Each userid can be used to sign on for any number of sessions on any CICS region. This is the compatibility option, providing the same signon scope as in releases of CICS before CICS Transaction Server for z/OS, Version 3 Release 1.

**CICS** Each userid can be signed on once only in the same CICS region. A signon request is rejected if the userid is already signed on to the same CICS region. However, the userid can be used to signon to another CICS region in the same, or another, MVS image.

#### **MVSIMAGE**

Each userid can be signed on once only, and to only one of the set of CICS regions in the same MVS image that also specify SNSCOPE=MVSIMAGE. A signon request is rejected if the user is already signed on to another CICS region in the same MVS image.

#### **SYSPLEX**

Each userid can be signed on once only, and to only one of the set of CICS regions within an MVS sysplex that also specify SNSCOPE=SYSPLEX. A signon is rejected if the user is already signed on to another CICS region in the same MVS sysplex.

The signon scope (if specified) applies to all userids signing on by an explicit signon request (for example, by an EXEC CICS SIGNON command or the CESN transaction). SNSCOPE is restricted to users signing on at local terminals, or signing on after using the CRTE transaction to connect to another system.

Signon scope specified by SNSCOPE *does not* apply to:

- Non-terminal users.
- The CICS default userid, specified by the DFLTUSER system initialization parameter.
- Preset userids, specified in the USERID option of the DEFINE TERMINAL command.
- Userids for remote users, received in attach headers.
- Userids for link security. For information about which userid is used for link security on a specific connection, see the *CICS RACF Security Guide*.
- The userid specified on the PLTPIUSR system initialization parameter.
- The CICS region userid.

**Restrictions** You can specify the SNSCOPE parameter in the SIT, PARM, or SYSIN only.

**SPCTR={ (1,2) | (1[,2][,3]) | ALL | OFF }**

specifies the level of tracing for all CICS components used by a transaction,

terminal, or both, selected for special tracing. If you want to set different tracing levels for an individual component of CICS, use the SPCTRxx system initialization parameter. For a list of all the available trace points and their level numbers, see *CICS Trace Entries*. For information about the differences between special and standard CICS tracing, see the *CICS Problem Determination Guide*.

It is possible to select up to 32 levels of tracing using the SPCTR system initialization parameter. However, most CICS components only use levels 1, 2 and 3, and some do not have trace points at all these levels. The exceptions are the SM component (storage manager domain), which also has level 4 tracing; and the SJ component (JVM domain), which also has trace levels 29–32, that are reserved to indicate the JVM trace levels 0, 1, and 2, plus a user-definable JVM trace level. You are recommended to use the SPCTRxx system initialization parameter, rather than the SPCTR system initialization parameter, to set special tracing levels above 3 for these components.

**number**

The level numbers for the level of special tracing you want for all CICS components. The normal options are: 1, (1,2), or (1,2,3). The default, (1,2), specifies special tracing for levels 1 and 2 for all CICS components.

**ALL** Enables the special tracing facility for all available levels.

**OFF** Disables the special tracing facility.

**SPCTRxx={ (1,2 ) | (1[,2] [,3] [,4] [,29] [,30] [,31] [,32]) | ALL | OFF }**

specifies the level of tracing for a particular CICS component used by a transaction, terminal, or both, selected for special tracing. You identify the component by coding a value for xx in the keyword. You code one SPCTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by SPCTRxx, the trace level is that set by SPCTR (which, in turn, defaults to (1,2)). The CICS component codes that you can specify for xx on the SPCTRxx keyword are shown in Table 24 on page 237:

Table 24. CICS component names and abbreviations

Code	Component name	Code	Component name
AP	Application domain	BA	Business application manager
BM	Basic mapping support	BR	3270 Bridge
CP	Common programming interface	DC	Dump compatibility layer
DD	Directory manager	DH	Document handler domain
DM	Domain manager domain	DP	Debugging profiles domain
DS	Dispatcher domain	DU	Dump domain
EI	Exec interface	EJ	Enterprise Java domain
EM	Event manager domain	FC	File control
GC	Global catalog domain	IC	Interval control
IE	ECI over TCP/IP domain	II	IIOp domain
IS	Inter-system communication	KC	Task control
KE	Kernel	LC	Local catalog domain
LD	Loader domain	LG	Log manager domain
LM	Lock manager domain	ME	Message domain
MN	Monitoring domain	NQ	Enqueue manager
OT	Object transaction services domain	PA	Parameter domain
PC	Program control	PG	Program manager domain
PI	Pipeline manager domain	PT	Partner domain
RI	Resource manager interface (RMI)	RM	Recovery manager
RX	RRMS domain	RZ	Request streams
SC	Storage control	SH	Scheduler services domain
SJ	JVM domain	SM	Storage manager domain
SO	Sockets domain	ST	Statistics domain
SZ	Front end programming interface	TC	Terminal control
TD	Transient data	TI	Timer domain
TR	Trace domain	TS	Temporary storage
UE	User exit interface	US	User domain
WB	WEB domain	XM	Transaction manager
XS	Security domain		

### number

The level numbers for the level of special tracing you want for the CICS component indicated by xx. Level numbers 1, 2, 3, 4, 29, 30, 31 and 32 can be used, depending on the component.

Most CICS components only use levels 1, 2 and 3, and some do not have trace points at all these levels. The exceptions are:

- The SM component (storage manager domain), which also has level 4 tracing. This level of tracing is intended for IBM field engineering staff.
- The SJ component (JVM domain), which also has trace levels 29–32, that are reserved to indicate the JVM trace levels 0, 1, and 2, plus a user-definable JVM trace level. You can use the system initialization

parameters JVMLEVEL0TRACE, JVMLEVEL1TRACE, JVMLEVEL2TRACE and JVMUSERTRACE to specify options for these JVM trace levels, and then activate them using the SPCTRSJ system initialization parameter.

**ALL** You want all the available levels of special CICS tracing switched on for the specified component.

**OFF** Switches off all levels of special CICS tracing for the CICS component indicated by xx.

For details of using trace, see the *CICS Problem Determination Guide*.

**Notes:**

1. The component codes BA, BM, CP, DC, DH, EI, FC, IC, IS, KC, PC, SC, SP, TC, TD, and UE are sub-components of the AP domain. As such, the corresponding trace entries are produced with a point ID of AP nnnn.
2. When you activate JVM trace, using trace levels 29–32 for the SJ component, the JVM trace appears as CICS trace point SJ 4D01.

**Restrictions** You can specify the SPCTRxx parameter in PARM, SYSIN, or CONSOLE only.

**SPOOL={NO|YES}**

specifies whether the system spooling interface is required.

**NO** The system spooling interface is not required.

**YES** The system spooling interface is required.

# The CICS pool interface uses the MVS exit, IEFDOIXT, which is provided in  
# the SYS1.LINKLIB library. For further information about the MVS exit IEFDOIXT,  
# see the current z/OS release information on z/OS MVS Installation Exits.

**SRBSVC={215|number}**

specifies the number that you have assigned to the CICS type 6 SVC. The default number is 215.

For information on changing the SVC number, see "Installing the CICS Type3 SVC" and "Selecting the high-performance option" in the *CICS Transaction Server for z/OS Installation Guide*. A CICS type 6 SVC with the specified (or default) number must have been link-edited with the system nucleus.

**SRT={1\$|YES|NO|xx}**

specifies the system recovery table suffix (see page "Defining CICS resource table and module keywords" on page 162.) For information about coding the macros for this table, see the *CICS Resource Definition Guide* manual.

If SRT=YES is coded, the default DFHSRT1\$ table is used.

If SRT=NO is coded, the system recovery program (DFHSRP) does not attempt to recover from a program check or from an operating system abend. However, CICS issues ESPIE macros to intercept program checks to perform clean-up operations before CICS terminates. Therefore, an SRT must be provided if recovery from either program checks or abnormal terminations, or both, is required.

**SRVERCP={037|codepage}**

Specifies the default server code page to be used by the DFHCNV data conversion table but only if the SRVERCP parameter in the DFHCNV macro is set to SYSDEF. The *codepage* is a field of up to 8 characters and can take the

values supported by the SRVERCP parameter in the DFHCNV macro. See the *CICS Family: Communicating from CICS on System/390* for the list of valid code pages. The default is 037.

**SSLCACHE={CICS | SYSPLEX}**

Specifies whether SSL is to use the local or sysplex caching of session ids. Sysplex caching is only allowed if multiple CICS socket-owning regions accept SSL connections at the same IP address.

**SSLDELAY={600 | number}**

Specifies the length of time in seconds for which CICS retains session ids for secure socket connections. Session ids are tokens that represent a secure connection between a client and an SSL server.

While the session id is retained by CICS within the SSLDELAY period, CICS can continue to communicate with the client without the significant overhead of an SSL handshake. The value is a number of seconds in the range 0 through 86400.

**SSLTCBS={8 | number}}**

This parameter is now obsolete and is only kept for compatibility. If it is specified, it is rejected with a message and MAXSSLTCBS is assumed.

**START=({AUTO | INITIAL | COLD | STANDBY} [, ALL])**

specifies the type of start for the system initialization program. The value specified for START, or the default of AUTO, becomes the default value for each resource.

**AUTO** CICS performs a warm, emergency, cold or initial start, according to the status of two control records on the global catalog:

- The recovery manager (RM) control record written by the previous execution of CICS
- The RM autostart override record written by a run of the recovery manager utility program, DFHRMUTL

**Note:** If the global catalog does *not* contain the RM control record:

- If it contains an RM autostart override record with option AUTOINIT, CICS performs an initial start.
- If it does not contain an RM autostart override record with option AUTOINIT, CICS does not start.

If you code START=AUTO, you must do one of the following:

- Provide the global catalog and system log from the previous execution of CICS. For an emergency restart to be successful, you must also have coded an activity keypoint value (see the AKPFREQ parameter on page “AKPFREQ” on page 166) on the previous execution of CICS.
- Provide a global catalog against which you have run the DFHRMUTL utility program, specifying SET\_AUTO\_START=AUTOINIT.

You may choose to leave the START parameter set to AUTO for all types of startup other than XRF standby, and use the DFHRMUTL program to reset the startup mode to COLD or INITIAL when necessary, using SET\_AUTO\_START=AUTOCOLD or SET\_AUTO\_START=AUTOINIT, respectively. For information about the DFHRMUTL utility program, see the *CICS Operations and Utilities Guide*.

## INITIAL

The status of CICS resource definitions saved in the global catalog at the previous shutdown is ignored, and all resource definitions are reinstalled, either from the CSD or CICS control tables.

You should rarely need to specify `START=INITIAL`; if you simply want to reinstall definitions of local resources from the CSD, use `START=COLD` instead.

Examples of times when an initial start is necessary are:

- When bringing up a new CICS system for the first time.
- After a serious software failure, when the system log has been corrupted.
- If the global catalog is cleared or initialized.
- When you want to run CICS with a dummy system log. (If the system log is defined as a dummy, it is ignored.)

## COLD

The status of CICS resource definitions saved in the global catalog at the previous shutdown is ignored, and all resource definitions (except those for the system log) are reinstalled, either from the CSD or CICS control tables.

Resynchronization information in the global catalog relating to remote systems or to RMI-connected resource managers is preserved. The CICS system log is scanned during startup, and information regarding unit of work obligations to remote systems, or to non-CICS resource managers (such as DB2) connected through the RMI, is preserved. (That is, any decisions about the outcome of local UOWs, needed to allow remote systems or RMI resource managers to resynchronize their resources, are preserved.)

Note that, on a cold start, the following are *not* preserved:

- Updates to *local* resources that were not fully committed or backed out during the previous execution, *even if the updates were part of a distributed unit of work*.
- Resynchronization information for remote systems connected by LU6.1 links, or for earlier releases of CICS systems connected by MRO.

If you want to reinstall resource definitions from the CSD, use `START=COLD` rather than `START=INITIAL`.

## STANDBY

Coding `START=STANDBY`, but only when you have also specified `XRF=YES`, defines this CICS as the alternate CICS region in an XRF pair. In other words, you **must** specify `START=STANDBY` for the system that starts off as the alternate. (To start an active CICS region, specify `AUTO` or `COLD`, as you would without XRF.)

## (option,ALL)

The ALL option is a special option you can use on the `START` parameter when you supply it as a system initialization parameter at CICS startup; you cannot code it in the SIT. If you specify `START=(AUTO,ALL)`, CICS initializes all resources according to the type of startup that it selects (warm, emergency, initial, or cold). The ALL option overrides any individual settings in other system initialization parameters.

However, if you do not use the ALL option, you can individually cold start those resources that have a COLD operand. For details of resources that have a COLD option, see Table 18 on page 162.

**Restrictions** You can specify START=(option,ALL) in PARM, SYSIN, or CONSOLE only.

For more information about the types of CICS startup, see “Controlling start and restart” on page 278.

**STARTER={NO|YES}**

specifies whether the generation of starter system modules (with \$ and # suffixes) is permitted, and various MNOTES are to be suppressed. This parameter should only be used when service is being performed on starter system modules.

**Restrictions** You can specify the STARTER parameter in the SIT only.

**STATEOD={0|hhmmss}**

specifies the end-of-day time in the format hhmmss. The default is 0, which is midnight.

End-of-day time is expressed in local time and must be in the range 00:00:00-23:59:59. That is, the hh value cannot exceed 23, and the mm and ss values can be specified in the range 00 to 59. If you leave out leading zeros, the DFHSIT macro inserts them (for example, 100 becomes 000100—that is, 1 minute 00 seconds past midnight).

This parameter is the equivalent of the ENDOFDAY option on the CEMT and EXEC CICS SET STATISTICS command, which you can use to modify the value set by STATEOD.

**STATINT={030000|hhmmss }**

specifies the recording interval for system statistics in the format hhmmss. The default is 3 hours.

The interval must be at least one minute and cannot be more than 24 hours. The minutes and seconds part of the value can be specified in the range 00 to 59. If you leave out leading zeros, the DFHSIT macro inserts them (for example, 3000 becomes 003000—that is, an interval of 30 minutes).

This parameter is the equivalent of the INTERVAL option on the CEMT and EXEC CICS SET STATISTICS command, which you can use to modify the value set by STATINT.

**STATRCD={OFF|ON}**

specifies the interval statistics recording status at CICS initialization. This status is recorded in the CICS global catalog for use during warm and emergency restarts. Statistics collected are written to the SMF data set.

**OFF** Interval statistics are not collected (no action is taken at the end of an interval).

End-of-day statistics are collected at the logical end of day and on shutdown. Unsolicited statistics are written to SMF as resources are discarded or closed.

**ON** Interval statistics are collected.

On a cold start of a CICS region, interval statistics are recorded by default at three-hourly intervals. All intervals are timed using the end-of-day time (midnight is the default) as a base starting time (**not**

CICS startup time). This means that the default settings give collections at 00.00, 03.00, 06.00, 09.00, and so on, regardless of the time that you start CICS.

On a warm or emergency restart the statistics recording status is restored from the CICS global catalog.

You can change the statistics recording status at any time as follows:

- During a warm or emergency restart by coding the STATRCD system initialization parameter.
- While CICS is running by using the CEMT or EXEC CICS SET STATISTICS command.

Whatever the value of the STATRCD system initialization parameter, you can ask for requested statistics and requested reset statistics to be collected. You can get statistics “on demand” for all, or for specified, resource types by using the CEMT or EXEC CICS PERFORM STATISTICS command. The period covered for statistics requested in this way is from the last reset time (that is, from the beginning of the current interval or from when you last issued a CEMT or EXEC CICS statistics command specifying RESETNOW) up to the time that you issue the PERFORM STATISTICS command.

For information about using these CEMT commands, see *CICS Supplied Transactions*. For programming information about the EXEC CICS PERFORM commands, see the *CICS System Programming Reference* manual. For information about the statistics utility program DFHSTUP, or recording statistics in the sample program *hlq.SAMPLIB*, see the *CICS Operations and Utilities Guide*. For information about the sample programs, see the *CICS Operations and Utilities Guide*.

**STGPROT={NO|YES}**

specifies whether you want storage protection in the CICS region. The permitted values are NO (the default), or YES:

**NO** If you specify NO, or allow this parameter to default, CICS does not operate any storage protection, and runs in a single storage key as in earlier releases. See Table 31 on page 357 for a summary of how STGPROT=NO affects the storage allocation for the dynamic storage areas.

**YES** If you specify YES, and if you have the required hardware and software, CICS operates with storage protection, and observes the storage keys and execution keys that you specify in various system and resource definitions. See Table 31 on page 357 for a summary of how STGPROT=YES affects the storage allocation for the dynamic storage areas.

If you do not have the required hardware and software support, CICS issues an information message during initialization, and operates without storage protection.

**STGRCVY={NO|YES}**

specifies whether CICS should try to recover from a storage violation.

**NO** CICS does not try to repair any storage violation that it detects.

**YES** CICS tries to repair any storage violation that it detects.

In both cases, CICS continues unless you have specified in the dump table that CICS should terminate.

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 checks the check zones and abends the task if either has been overwritten.

Terminal input-output areas (TIOAs) have similar check zones, which are set up with identical values. At each freemain of a TIOA, CICS checks the check zones and abends the task if they are not identical.

If you specify STGRVCY(YES), CICS resets the check zones correctly and the task continues running.

#  
#  
#  
#

If you specify STGRVCY(NO), CICS abends the task if it is still running. The storage is not reusable and is not returned to the DSA for the remainder of the CICS cycle. If an error is detected when the task ends, no abend is issued. Any sync point that has taken place could save data that is corrupted.

**STNTR**={1 | (1[,2] [,3]) | ALL | OFF}

specifies the level of standard tracing required for CICS as a whole.

It is possible to select up to 32 levels of tracing using the STNTR system initialization parameter. However, most CICS components only use levels 1, 2 and 3, and some do not have trace points at all these levels. The exceptions are the SM component (storage manager domain), which also has level 4 tracing; and the SJ component (JVM domain), which also has trace levels 29–32, that are reserved to indicate the JVM trace levels 0, 1, and 2, plus a user-definable JVM trace level. You should use the STNTRxx system initialization parameter, rather than the STNTR system initialization parameter, if you need to set standard tracing levels above 3 for these components.

**Note: Warning!** Before globally activating tracing levels 3 and ALL, which will set these tracing levels for the storage manager (SM) component and the JVM domain (SJ) component, read the warnings given in the description for the STNTRxx system initialization parameter.

**number**

Code the level number(s) for the level of standard tracing you want for all CICS components. The options are: 1, (1,2), or (1,2,3). The default, 1, specifies standard tracing for level 1 for all CICS components.

**ALL** Enables standard tracing for all levels.

**OFF** Disables standard tracing.

For information about the differences between special and standard CICS tracing, see the *CICS Problem Determination Guide*.

**STNTRxx**={1 | (1[,2] [,3] [,4] [,29] [,30] [,31] [,32]) | ALL | OFF}

specifies the level of standard tracing you require for a particular CICS component. You identify the component by coding a value for xx in the keyword. You code one STNTRxx keyword for each component you want to define selectively. For a CICS component being specially traced that does not have its trace level set by STNTRxx, the trace level is that set by STNTR (which, in turn, defaults to 1). You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

The CICS component codes that you can specify for xx on this STNTRxx keyword are shown in Table 24 on page 237.

**ALL** You want all the available levels of standard tracing switched on for the specified component.

# **Warning!** Selecting ALL for standard tracing for the storage manager (SM) component, or the temporary storage domain (TS), degrades the performance of your CICS region. This is because ALL switches on trace flags that are used by SM domain for field engineering purposes.

#

#

#

# **Warning!** Selecting ALL for standard tracing for the JVM domain (SJ) component is not recommended. JVM trace can produce a large amount of output, so you should normally activate JVM trace for special transactions (using the SPCTRSJ system initialization parameter), rather than turning it on globally for all transactions.

#

#

**number**

The level number(s) for the level of standard tracing you want for the CICS component indicated by xx. Level numbers 1, 2, 3, 4, 29, 30, 31 and 32 can be used, depending on the component.

Most CICS components only use levels 1, 2 and 3, and some do not have trace points at all these levels. The exceptions are:

- The SM component (storage manager domain), which also has level 4 tracing. This level of tracing is intended for IBM field engineering staff.

**Warning!** Selecting tracing levels 3, 4, or ALL for standard tracing for the storage manager (SM) component, or the temporary storage domain (TS), degrades the performance of your CICS region. This is because options 3 and 4 (and ALL) switch on trace flags that are used by SM domain for field engineering purposes.

SM trace flag 3 deactivates the quickcell mechanism, and SM trace flag 4 forces subpool element chaining on every CICS subpool. Furthermore, once these settings have been activated during system initialization, they cannot be unset, either through a PLTPI program or by using the CETR trace transaction, because they are not used for tracing as such. Thus, a significant performance overhead is incurred if these storage manager trace levels are selected for standard tracing.

See the *CICS Problem Determination Guide* for information about the effects of trace levels 3 and 4.

- The SJ component (JVM domain), which also has trace levels 29–32, that are reserved to indicate the JVM trace levels 0, 1, and 2, plus a user-definable JVM trace level. You can use the system initialization parameters JVMLEVEL0TRACE, JVMLEVEL1TRACE, JVMLEVEL2TRACE and JVMUSERTRACE to specify options for these JVM trace levels, and then activate them using the SPCTRSJ system initialization parameter.

**Warning!** Selecting tracing levels 29, 30, 31, 32 or ALL for standard tracing for the JVM domain (SJ) component is not recommended. JVM trace can produce a large amount of output, so you should normally activate JVM trace for special transactions (using the SPCTRSJ system initialization parameter), rather than turning it on globally for all transactions.

**OFF** Switches off all levels of standard CICS tracing for the CICS component indicated by xx.

**Restrictions** You can specify the STNTRxx parameter in PARM, SYSIN, or CONSOLE only.

**SUBTSKS={0|1}**

specifies the number of task control blocks (TCBs) you want CICS to use for running tasks in concurrent mode. A concurrent mode TCB allows CICS to perform management functions as system subtasks.

CICS always uses at least two TCBs:

1. The quasi-reentrant mode TCB. CICS runs all user applications under this TCB.
2. The resource-owning mode TCB. CICS runs tasks that open and close files under this TCB.

If you specify SUBTSKS=0, CICS runs under these two TCBs.

If you specify SUBTSKS=1, CICS uses an additional TCB, a concurrent mode TCB, to perform system subtasking functions.

**SUFFIX=xx**

specifies the last two characters of the name of this system initialization table.

The first 6 characters of the name of the SIT are fixed as DFHSIT. You can specify the last two characters of the name, using the SUFFIX parameter. Because the SIT does not have a TYPE=INITIAL macro statement like other CICS resource control tables, you specify its SUFFIX on the TYPE=CSECT macro statement.

The suffix allows you to have more than one version of the SIT. Any one or two characters (other than NO and DY) are valid. You select the version of the table to be loaded into the system during system initialization by coding SIT=xx, either in the PARM parameter or the SYSIN data set. (You can, in some circumstances, specify the SIT using the system console, but this is not recommended.)

**Restrictions** You can specify the SUFFIX parameter in the SIT only.

**SYDUMAX={999|number}**

specifies the limit on the number of system dumps that can be taken per dump table entry. If this number is exceeded, subsequent system dumps for that particular entry will be suppressed. The SYDUMAX parameter applies for new or added system dump codes. It does not override the limit on the number of system dumps for existing dump table entries.

**number**

A number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

**SYSIDNT={CICS|name}**

specifies a 1-to 4-character name that is known only to your CICS region. If your CICS region also communicates with other CICS regions, the name you choose for this parameter to identify your local CICS region must not be the same name as an installed CONNECTION resource definition for a remote region.

The value for SYSIDNT, whether specified in the SIT or as an override, can only be updated on a cold start. After a warm start or emergency restart, the value of SYSIDNT is that specified in the last cold start.

For information about the SYSIDNT of a local CICS region, see the *CICS Intercommunication Guide*.

**SYSTR={ON|OFF}**

specifies the setting of the master system trace flag.

**ON** The master trace flag is set, causing CICS to write trace entries of system activity for the individual CICS components. Trace entries are captured and written only for those components for which the trace level is 1 or greater, as specified on the STNTR or STNTRxx system initialization parameters. Entries are written only to those trace destinations that are active.

**OFF** The master trace flag is unset, and no standard trace entries are captured, overriding any trace levels specified by the STNTR or STNTRxx system initialization parameters.

**Note:** Setting the master trace flag OFF affects only standard tracing and has no effect on special tracing, which is controlled separately by SPCTR or SPCTRxx trace levels and the CETR transaction.

See the *CICS Problem Determination Guide* for more information about controlling CICS trace.

**TAKEOVR={MANUAL|AUTO|COMMAND} (alternate)**

Use this parameter in the SIT for an alternate CICS region. It specifies the action to be taken by the alternate CICS region, following the (apparent) loss of the surveillance signal in the active CICS region. In doing this, it also specifies the level of operator involvement.

If both active and alternate CICS regions are running under different MVS images in the same sysplex, and an MVS failure occurs in the MVS image of the active CICS region, the TAKEOVR option is overridden.

- If the MVS images are running in a PR/SM environment, CICS XRF takeover to an alternate CICS region on a separate MVS image completes without the need for any operator intervention.
- If the MVS images are not running in a PR/SM environment, the CICS takeover is still initiated automatically, but needs operator intervention to complete, because XCF outputs a WTOR (IXC402D). Sysplex partitioning does not complete until the operator replies to this message, and CICS waits for sysplex partitioning to complete before completing the XRF takeover.

**MANUAL**

The operator is asked to approve a takeover if the alternate CICS region cannot detect the surveillance signal of the active CICS region.

The alternate CICS region does not ask the operator for approval if the active CICS region signs off abnormally, or if there is an operator or program command for takeover. In these cases, there is no doubt that the alternate CICS region should take over, and manual involvement by the operator would be an unnecessary overhead in the takeover process.

You could use this option, for instance, to ensure manual takeover of a master or coordinator region in MRO.

**AUTO** No operator approval, or intervention, is needed for a takeover.

**COMMAND**

Takeover occurs only when a CEBT PERFORM TAKEOVER command is received by the alternate CICS region. It ensures, for instance, that a dependent alternate CICS region, in MRO, is activated only if it receives the command from the operator, or from a master or coordinator region.

**TBEXITS=([name1] [,name2] [,name3] [,name4] [,name5] [,name6])**

specifies the names of your backout exit programs for use during emergency restart backout processing. For more information about backout exit programs, see the *CICS Customization Guide* and the *CICS Recovery and Restart Guide*.

The order in which you code the names is significant. If you do not want to use all the exits, code commas in place of the names you omit. For example:

```
TBEXITS=(, ,EXITF,EXITV)
```

The program names for *name1* through *name6* apply to global user exit points as follows:

- *name1* and *name2* are the names of programs to be invoked at the XRCINIT and XRCINPT global user exit points (but note that XRCINIT and XRCINPT are invoked only for user log records).
- *name3* is the name of the program to be invoked at the file control backout failure global user exit point, XFCBFAIL.
- *name4* is the name of the program to be invoked at the file control logical delete global user exit point, XFCLDEL.
- *name5* is the name of the program to be invoked at the file control backout override global user exit point, XFCBOVER.
- *name6* is the name of the program to be invoked at the file control backout override global user exit point, XFCBOUT.

This exit is invoked (if required) during backout of a unit of work, regardless of whether the backout is taking place at emergency restart, or at any other time.

The XFCBFAIL, XFCLDEL, and XFCBOVER global user exit programs are enabled on all types of CICS start if they are named on the TBEXITS system initialization parameter.

If no backout exit programs are required, you can do one of the following:

- Omit the TBEXITS system initialization parameter altogether
- Code the parameter as TBEXITS=(,,,,,)

**TCAM={NO|YES}**

This parameter is now obsolete and is only kept for compatibility. If it is specified, it is rejected with a message and TCAM=NO is assumed.

**TCP={YES|NO}**

specifies whether the pregenerated non-VTAM terminal control program, DFHTCP, is to be included.

You must code TCP=YES if you intend using card reader/line printer (sequential) devices.

**TCPIP={NO|YES}**

specifies whether CICS TCPIP services are to be activated at CICS startup. The default is NO, meaning that these services cannot be enabled. If TCPIP is set to YES, the HTTP, IIOP, and ECI over TCP/IP services can process work.

**Note:** The TCPIP system initialization parameter affects only CICS internal TCP/IP Services defined by TCPIPSERVICE resource definitions. It has nothing to do with the TCP/IP Socket Interface for CICS feature of TCP/IP for MVS.

**TCSACTN={NONE|UNBIND|FORCE}**

specifies the required action that CICS terminal control should take if the terminal control shutdown wait threshold expires. For details of the wait threshold, see the TCSWAIT system initialization parameter. TCSACTN only takes effect when TCSWAIT is coded with a value in the range 1 through 99.

This is a global default action. On a terminal-by-terminal basis, you can code a DFHZNEP routine to override this action.

**NONE** No action is taken. This can be overridden by DFHZNEP.

- To report hung terminals and not attempt to force-close them specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters.
- To attempt to force-close some hung terminals, and only report others, specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=NONE system initialization parameters, and code a DFHZNEP routine that selects the required terminals and sets TWAOCN on for them.

### **UNBIND**

CICS terminal control attempts to close the session by issuing a VTAM CLSDST and sending an SNA UNBIND command to the hung terminal. This can be overridden by DFHZNEP.

- To attempt to force-close all hung terminals specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=UNBIND system initialization parameters.

### **FORCE**

CICS terminal control attempts to forceclose the CICS VTAM ACB if there are any hung terminals or parallel connection sessions. All CICS VTAM terminals and sessions are released and CICS normal shutdown continues. This parameter will only take effect if all LU Type 6.2 parallel connections, if any, have successfully completed CNOS close processing.

- To attempt to force-close the CICS VTAM ACB if there are any hung terminals, specify the TCSWAIT=mm (with an appropriate time interval) and TCSACTN=FORCE system initialization parameters.

### **TCSWAIT={4|number|NO|NONE|0}**

specifies the required CICS terminal control shutdown wait threshold. The wait threshold is the time, during shutdown, that CICS terminal control allows to pass before it considers terminal shutdown to be hung. If all VTAM sessions shutdown and close before the threshold expires then the CICS shutdown process moves on to its next stage, and the terminal control wait threshold then no longer applies. If, however, some of the VTAM sessions do not complete shutdown and close, then CICS takes special action with these sessions. For details of this special action see the description of the TCSACTN system initialization parameter. The wait threshold only applies to VTAM sessions; that is, VTAM terminals and VTAM intersystem connections. The wait time is specified as a number of minutes, in the range 1 through 99. As a special case, TCSWAIT=NO may be specified to indicate that terminal control shutdown is never to be considered hung, no matter how long the shutdown and close process takes. TCSWAIT=NONE and TCSWAIT=0 are alternative synonyms for TCSWAIT=NO, and all three have the same effect (internally they are held as the one value 0 (zero)).

The value that you specify on the TCSWAIT system initialization parameter should be large enough so that under normal circumstances all VTAM terminals and connections shutdown in an orderly fashion. To help choose this value, consider using a value slightly larger than the elapsed time between the following two CICS terminal control shutdown messages:

```
DFHZC2305 Termination of VTAM sessions beginning
DFHZC2316 VTAM ACB is closed
```

**TCT={NO|xx|YES}**

specifies which terminal control table, if any, is to be loaded. (See page “Defining CICS resource table and module keywords” on page 162.) For guidance about coding the macros for this table, see the *CICS Resource Definition Guide*

If you reassemble the TCT after starting CICS, any changes are applied when you next start CICS, even if it is a warm or emergency startup.

If you have VTAM-connected terminals only, you can specify TCT=NO. If you do this, note that a dummy TCT, called DFHTCTDY, is loaded during system initialization. For more information about DFHTCTDY, see page “The dummy TCT, DFHTCTDY” on page 271. (If you code TCT=NO, you must specify a CSD group list in the GRPLIST parameter.)

**TCTUAKEY={USER|CICS}**

specifies the storage key for the terminal control table user areas (TCTUAs) if you are operating CICS with storage protection (STGPROT=YES). The permitted values are USER (the default), or CICS:

**USER** CICS obtains the amount of storage for TCTUAs in user key. This allows a user program executing in any key to modify the TCTUA.

**CICS** CICS obtains the amount of storage in CICS key. This means that only programs executing in CICS key can modify the TCTUA, and user-key programs have read-only access.

If CICS is running without storage protection, the TCTUAKEY parameter only designates which DSA (User or CICS) the storage comes from. The TCTUAs are accessed in CICS-key whether they are in the UDSA or CDSA.

See “The terminal control table user areas” on page 355 for more information about TCTUAs.

**TCTUALOC={BELOW|ANY}**

specifies where terminal user areas (TCTUA) are to be stored.

**BELOW** The TCTUAs are stored below the 16MB line.

**ANY** The TCTUAs are stored anywhere in virtual storage. CICS stores TCTUAs above the 16MB line if possible.

For more information about TCTUAs, see “Accessing the CSD by the offline utility program, DFHCSDUP” on page 79.

For details about defining terminals using RDO, see the *CICS Resource Definition Guide*.

**TD=({3|decimal-value-1}[,{ 3|decimal-value-2}])**

specifies the number of VSAM buffers and strings to be used for intrapartition transient data (TD).

**decimal-value-1**

The number of buffers to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 32 767. The default value is 3.

CICS obtains, above the 16MB line, storage for the TD buffers in units of the page size (4KB). Because CICS optimizes the use of the storage obtained, TD may allocate more buffers than you specify, depending on the control interval (CI) size you have defined for the intrapartition data set.

For example, if the CI size is 1536, and you specify 3 buffers (the default number), CICS actually allocates 5 buffers. This is because 2

pages (8192 bytes) are required to obtain sufficient storage for three 1536-byte buffers, a total of only 4608 bytes, which would leave 3584 bytes of spare storage in the second page. In this case, CICS allocates another 2 buffers (3072 bytes) to minimize the amount of unused storage. In this way CICS makes use of storage that would otherwise be unavailable for any other purpose.

#### **decimal-value-2**

The number of VSAM strings to be allocated for the use of intrapartition transient data. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TD=(8,5) specifies 8 buffers and 5 strings.

The operands of the TD parameter are positional. You must code commas to indicate missing operands if others follow. For example, TD=(,2) specifies the number of strings and allows the number of buffers to default.

#### **TDINTRA={NOEMPTY | EMPTY}**

specifies whether CICS is to initialize with empty intrapartition TD queues.

##### **NOEMPTY**

CICS recovers all the intrapartition TD queues to the state they were in at the previous termination of CICS, as in a normal emergency restart. The TD queue resource definitions are recovered from the CICS global catalog.

##### **EMPTY**

CICS initializes with all the intrapartition TD queues empty. This option must be used when CICS is initializing in remote site recovery mode (OFFSITE=YES).

You can optionally use this option to COLD start your intra-partition TD queues to initialize them as empty.

The option is significant only on warm and emergency restarts—cold starts always initialize with empty queues. Note that the EMPTY option may cause data integrity problems because all in-doubt log records associated with logically recoverable TD queues are discarded.

The TD queue resource definitions are recovered from the CICS global catalog.

#  
#  
#  
#  
#  
#

#### **TDSUBTASK={OFF | ON}**

specifies whether CICS should use the FO TCB to write to an extrapartition transient data queue, where the record format is FIXED and the block format is UNBLOCKED. The default is OFF, so that no TCB switch can occur. This particularly benefits you if you are submitting work to JES using the internal reader (INTRDR).

#### **TRANISO={NO | YES}**

specifies, together with the STGPROT system initialization parameter, whether you want transaction isolation in the CICS region. The permitted values are NO (the default), or YES.

##### **NO**

This is the default. If you specify NO, or allow this parameter to default, CICS operates without transaction isolation, and all storage in the CICS address space is addressable. If you specify STGPROT=YES and TRANISO=NO, CICS storage protection is active without transaction isolation.

**YES** Transaction isolation is required. If you specify TRANISO=YES and STGPROT=YES, and you have the required hardware and software, CICS operates with transaction isolation. This ensures that the user-key task-lifetime storage of transactions defined with the ISOLATE(YES) option is isolated from the user-key programs of other transactions.

If you specify TRANISO=YES, but you do not have the required hardware and software or STGPROT=NO is specified, CICS issues an information message during initialization, and operates without transaction isolation.

STGPROT=NO and TRANISO=YES specified in the system initialization table causes an error during assembly (MNOTE 8).

**Note:** VSAM nonshared resources (NSR) are not supported for transactions that use transaction isolation. You should specify ISOLATE(NO) when you define transactions that access VSAM files using NSR.

**TRAP={OFF|ON}**

specifies whether the FE global trap exit is to be activated at system initialization. This exit is for diagnostic use under the guidance of service personnel. For background information about this exit, see the *CICS Problem Determination Guide*.

**TRDUMAX={999|number}**

specifies the limit on the number of transaction dumps that may be taken per Dump Table entry. If this number is exceeded, subsequent transaction dumps for that particular entry will be suppressed.

**number**

A number in the range 0 through 999. The default, 999, enables an unlimited number of dumps to be taken.

**TRTABSZ={16|number-of-kilobytes}**

specifies the size in kilobytes of the internal trace table. (1KB = 1024 bytes.) The CICS trace table is allocated in virtual storage above the 16MB line, and it is allocated **before** the extended CICS-key DSA (ECDSA) and the extended user-key DSA (EUDSA). Ensure that there is sufficient virtual storage for the trace table, the ECDSA, and the EUDSA by specifying a large enough region size on the MVS REGION parameter of your CICS job.

#  
#  
#  
#

Use caution when setting this parameter to a very high value. There must be enough MVS page storage to satisfy both the request as well as the DSA sizes. Use the DISPLAY ASM MVS system command to display current information about the status and utilization of all MVS page data sets.

**16** 16KB is the default size of the trace table, and also the minimum size.

**number**

The number of kilobytes of storage to be allocated for the internal trace table, in the range 16KB through 1048576KB. Subpool 1 is used for the trace table storage, which exists for the duration of the CICS execution. The table is page aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4KB), it is rounded up to the next multiple of 4KB.

Trace entries are of variable lengths, but the average length is approximately 100 bytes.

**Note:** To switch on internal tracing, use the INTTR parameter; for a description of INTTR, see page "INTTR" on page 205.

**TRTRANSZ={16|number-of-kilobytes}**

specifies the size in kilobytes of the transaction dump trace table. (1KB = 1024 bytes.)

When a transaction dump is taken, CICS performs an MVS GETMAIN for storage above the 16MB line for the transaction dump trace table.

**16** 16KB is the default size of the transaction dump trace table.

**number**

The number of kilobytes of storage to be allocated for the transaction dump trace table, in the range 16KB through 1048576KB.

**TRTRANTY={TRAN|ALL}**

specifies which trace entries should be copied from the internal trace table to the transaction dump trace table.

**TRAN** Only the trace entries associated with the transaction that is abending will be copied to the transaction dump trace table.

**ALL** All of the trace entries from the internal trace table will be copied to the transaction dump trace table. If the internal trace table size is larger than the transaction dump trace table size, the transaction dump trace table could wrap. This results in only the most recent trace entries being written to the transaction dump trace table.

**TS=([COLD] [, {0|3|decimal-value-1 } ] [, {3|decimal-value-2} ])**

specifies:

- Whether you want to cold start temporary storage.
- The number of VSAM buffers to be used for auxiliary temporary storage.
- The number of VSAM strings to be used for auxiliary temporary storage.

**COLD**

The type of start for the temporary storage facility. COLD forces a cold start regardless of the value of the START parameter. If COLD is omitted, the TS start type is determined by the value of START.

**0** No buffers are required; that is, only MAIN temporary storage is required.

**decimal-value-1**

The number of buffers to be allocated for the use of auxiliary temporary storage. The value must be in the range 3 through 32 767.

**decimal-value-2**

The number of VSAM strings to be allocated for the use of auxiliary temporary storage. The value must be in the range 1 through 255, and must not exceed the value specified in decimal-value-1. The default value is 3.

For example, TS=(,8,5) specifies 8 buffers and 5 strings.

The operands of the TS parameter are positional. You must code commas to indicate missing operands if others follow. For example, TS=(,8) specifies the number of buffers and allows the other operands to default.

**TST={NO|YES|xx}**

specifies the temporary storage table suffix. (See page “Defining CICS resource table and module keywords” on page 162.)

For information about coding the macros for this table, see the *CICS Resource Definition Guide*

**UDSASZE={0K|number}**

specifies the size of the UDSA. The default size is 0, indicating that the DSA size can change dynamically. A non-zero value indicates that the DSA size is fixed.

**number**

specify number as an amount of storage in the range 0 to 16777215 bytes in multiples of 262144 bytes (256KB). If the size specified is not a multiple of 256KB (or 1MB if transaction isolation is active), CICS rounds the value up to the next multiple.

You can specify number in bytes (for example, 4194304), or as a whole number of kilobytes (for example, 4096K), or a whole number of megabytes (for example, 4M).

**Restrictions** You can specify the UDSASZE parameter in PARM, SYSIN, or CONSOLE only.

**UOWNETQL=user\_defined\_value**

specifies a qualifier for the NETUOWID for units of work initiated on the local CICS region. UOWNETQL is required only if VTAM=NO is coded. The specified value is used in the following circumstances:

- CICS is being cold started and VTAM=NO has been specified.
- CICS is being cold started and the VTAM ACB has failed to open.
- CICS is being started with VTAM=NO and the VTAM ACB has not been opened since CICS was last cold started.
- CICS is being started, the VTAM ACB has failed to open, and the VTAM ACB has not been opened since CICS was last cold started.

If any of the above conditions apply and UOWNETQL is not specified, a dummy default UOWNETQL of 9UNKNOWN is used. This dummy UOWNETQL is invalid because the first character is a number. UOWNETQL is given this invalid name to avoid a conflict with any real, valid netid.

The value you code can be from 1 to 8 characters long, and must consist of uppercase letters (A through Z), or numbers in the range 0 through 9. The first character must be a letter.

**USERTR={ON|OFF}**

specifies whether the master user trace flag is to be set on or off. If the user trace flag is off, the user trace facility is disabled, and EXEC CICS ENTER TRACENUM commands receive an INVREQ condition if EXCEPTION is not specified. If the program does not handle this condition the transaction will abend AEIP.

For programming information about the user trace facility using EXEC CICS ENTER TRACENUM commands, see the *CICS Application Programming Reference* manual.

**USRDELAY={30|number}**

specifies the maximum time, in the range 0 through 10080 minutes (up to 7 days), that an eligible userid and its associated attributes are to be retained in the user table if the userid is unused. An entry in the user table for a userid that is retained during the delay period can be reused.

The userids eligible for reuse within the USRDELAY period are any that are:

- Received from remote systems.
- Specified on SECURITYNAME in CONNECTION definitions.
- Specified on USERID in SESSIONS definitions.
- Specified on USERID in the definition of an intrapartition transient data queue.
- Specified on USERID on START commands.
- Specified on USERID for non-terminal tasks, such as the alias tasks that are attached for processing HTTP requests.

#  
#

Within the USRDELAY period, a userid in any one of these categories can be reused in one of the other categories, provided the request for reuse is qualified with the same qualifiers. If a userid is qualified by a different group id, APPLID, or terminal id, a retained entry is **not** reused (except when changing the terminal ID on LU6.2 when the retained entry **is** used).

If a userid is unused for more than the USRDELAY limit, it is removed from the system, and the message DFHUS0200 is issued. You can suppress this message in an XMEOUT global user exit program. If you specify USRDELAY=0, all eligible userids are deleted immediately after use, and the message DFHUS0200 is not issued. Do not code USRDELAY=0 if this CICS region communicates with other CICS regions and:

- ATTACHSEC=IDENTIFY is specified on the CONNECTION definitions for the connections used,  
and
- The connections used carry high volumes of transaction routing or function shipping activity.

You should specify a value that gives the optimum level of performance for your CICS environment.

If you specify USRDELAY=0 in the above scenario, CICS drives a full signon for each incoming request (with I/O to RACF) and a full signoff at the end of each transaction. For function shipping, there may be multiple signons/signoffs driven on a data-owning region for one task on an AOR.

**Note:** If a value, other than 0, is specified for USRDELAY, the ability to change the user's attributes or revoke the userid becomes more difficult because the userid and its attributes are retained in the region until the USRDELAY value has expired. For example, if you have specified USRDELAY=30 for a userid, but that userid continues to run transactions every 25 minutes, the USRDELAY value will never expire and any changes made to the userid will never come into effect.

When running a remote transaction, a userid remains signed-on to the remote CICS region (after the conversation associated with the first attach request is complete) until the delay specified by USRDELAY has elapsed since the last transaction associated with the attach request for the userid has completed. When this event occurs, the userid is removed from the remote CICS region.

For more information about the use of USRDELAY, see the *CICS Performance Guide*.

**VTAM={YES|NO}**

specifies whether the VTAM access method is to be used. The default is VTAM=YES.

**VTPREFIX={\|character}**

specifies the first character to be used for the terminal identifiers (termids) of autoinstalled virtual terminals. Virtual terminals are used by the External Presentation Interface (EPI) and terminal emulator functions of the CICS Client products.

Termids generated by CICS for autoinstalled Client terminals consist of a 1-character prefix and a 3-character suffix. The default prefix is '\'. The suffix can have the values 'AAA' through '999'. That is, each character in the suffix can have the value 'A' through 'Z' or '0' through '9'. The first suffix generated by CICS has the value 'AAA'. This is followed by 'AAB', 'AAC', ... 'AAZ', 'AA0', 'AA1', and so on, up to '999'.

Each time a Client virtual terminal is autoinstalled, CICS generates a 3-character suffix that it has not recorded as being in use.

By specifying a prefix, you can ensure that the termids of Client terminals autoinstalled on this system are unique in your transaction routing network. This prevents the conflicts that could occur if two or more terminal-owning regions (TORs) ship definitions of Client virtual terminals to the same application-owning region (AOR).

If such a naming conflict does occur—that is, if a Client virtual terminal is shipped to an AOR on which a remote terminal of the same name is already installed—the autoinstall user program is invoked in the AOR. Your user program can resolve the conflict by allocating an alias terminal identifier to the shipped definition. (For details of writing an autoinstall user program to install shipped definitions, see the *CICS Customization Guide*.) However, you can avoid potential naming conflicts by specifying a different prefix, reserved for virtual terminals, on each TOR on which Client virtual terminals are to be installed.

You must not use the characters + - \* < > = { } or blank.

**Notes:**

1. When specifying a prefix, ensure that termids generated by CICS for Client terminals do not conflict with those generated by your autoinstall user program for user terminals, or with the names of any other terminals or connections.
2. Client terminal definitions are not recovered after a restart. Immediately after a restart, no Client terminals are in use, so when CICS generates suffixes it begins again with 'AAA'. This means that CICS does **not** always generate the same termid for any given Client terminal. This in turn means that server applications should not assume that a particular CICS-generated termid always equates to a particular Client terminal.

If your server programs do make this assumption, you can use your autoinstall user program to allocate alias termids, by which the virtual terminals will be known to CICS, in a consistent manner.

3. Clients can override CICS Transaction Server for z/OS-generated termids.

For further information about Client virtual terminals, see the *CICS Intercommunication Guide* manual.

**WEBDELAY**=(5|time\_out,60|keep\_time)

Specifies two Web delay periods:

1. A time-out period. The maximum time, in minutes, in the range 1-60, that a transaction started through the Web 3270 bridge interface, is allowed to remain in terminal wait state before it is automatically purged by CICS.
2. The terminal keep time. The time, in minutes, in the range 1-6000, during which state data is kept for a CICS Web 3270 bridge transaction, before CICS performs clean-up.

**WRKAREA**={512|number}

specifies the number of bytes to be allocated to the common work area (CWA). This area, for use by your installation, is initially set to binary zeros, and is available to all programs. It is not used by CICS. The maximum size for the work area is 3584 bytes.

**XAPPC**={NO|YES}

specifies whether RACF session security can be used when establishing APPC sessions.

**NO** RACF session security cannot be used.

**YES** RACF session security can be used.

If you specify BINDSECURITY=YES for a particular APPC connection, a request to RACF is issued to extract the security profile. If the profile exists, it is used to bind the session.

**Note:** If you specify XAPPC=YES, the external security manager that you use must support the APPCLU general resource class, otherwise CICS fails to initialize.

**Restrictions** You can specify the XAPPC parameter in the SIT, PARM, or SYSIN only.

**XCMD**={YES|name|NO}

specifies whether you want CICS to perform command security checking, and optionally the RACF resource class name in which you have defined the command security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. Such checking is performed every time a transaction tries to use a COLLECT, DISABLE, DISCARD, ENABLE, EXTRACT, INQUIRE, PERFORM, RESYNC, or SET command, or any of the FEPI commands, for a resource.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the CMDSEC(YES) option on the transaction resource definition.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF, using the default class name of CICSCMD prefixed by C or V, to check whether the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is CCICSCMD and the grouping class name is VCICSCMD.

**name** CICS calls RACF, using the specified resource class name prefixed by C or V, to verify that the userid associated with a transaction is authorized to use a CICS command for the specified resource. The resource class name is *Cname* and the grouping class name is *Vname*.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any command security checks, allowing any user to use commands that would be subject to those checks.

**Restrictions** You can specify the XCMD parameter in the SIT, PARM, or SYSIN only.

**XDB2={NO|name}**

specifies whether you want CICS to perform DB2ENTRY security checking.

**NO** CICS does not perform any DB2 resource security checks.

**name** CICS calls RACF, using the specified general resource class name, to check whether the userid associated with the CICS DB2 transaction is authorized to access the DB2ENTRY referenced by the transaction.

Unlike the other *Xaaa* system initialization parameters, this DB2 security parameter does not provide a YES option that implies a default CICS resource class name for DB2ENTRY resources. You have to specify your own DB2 resource class name.

**XDCT={YES|name|NO}**

specifies whether you want CICS to perform transient data resource security checking. If you specify YES or a RACF resource class name, CICS calls RACF to verify that the userid associated with a transaction is authorized to access the transient data destination. Such checking is performed every time a transaction tries to access a transient data destination.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the transaction resource definition.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF, with the default CICS resource class name of CICS DCT prefixed by D or E, to verify whether the userid associated with the transaction is authorized to access the specified destination.

The resource class name is DCICSDCT and the grouping class name is ECICSDCT.

**name** CICS calls RACF, using the specified resource class name, to check whether the userid associated with the transaction is authorized to access the specified destination. The resource class name is *Dname* and the grouping class name is *Ename*.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any transient data security checks, allowing any user to access any transient data destination.

**Restrictions** You can specify the XDCT parameter in the SIT, PARM, or SYSIN only.

**XEJB={YES|NO}**

specifies whether support of security roles is enabled.

**YES** CICS support for security roles is enabled:

- When an application invokes a method of an enterprise bean, CICS calls the external security manager to verify that the userid associated with the transaction is defined in at least one of the security roles associated with the method.
- When an application invokes the `isCallerInRole()` method, CICS calls the external security manager to determine whether the userid associated with the transaction is defined in the role specified on the method call.

**NO** CICS support for security roles is disabled:

- CICS does not perform enterprise bean method level checks, allowing any userid to invoke any enterprise bean method.
- The `isCallerInRole()` method always returns a value of true.

**Restrictions:**

1. You can specify the XEJB parameter in the SIT, PARM, or SYSIN only.
2. To enable security role support, you must also specify SEC=YES.

**XFCT={YES|name|NO}**

specifies whether you want CICS to perform file resource security checking, and optionally specifies the RACF resource class name in which you have defined the file resource security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with a transaction is authorized to access File Control-managed files. Such checking is performed every time a transaction tries to access a file managed by CICS File Control.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the resource definitions.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF, using the default CICS resource class name of CICSFCT prefixed by F or H, to verify that the userid associated with a transaction is authorized to access files reference by the transaction. The resource class name is FCICSFCT and the grouping class name is HCICSFCT.

**name** CICS calls RACF, using the specified resource class name, to verify that the userid associated with a transaction is authorized to access files referenced by the transaction. The resource class name is *Fname* and the grouping class name is *Hname*.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any file resource security checks, allowing any user to access any file.

**Restrictions** You can specify the XFCT parameter in the SIT, PARM, or SYSIN only.

**XJCT={YES|name|NO}**

specifies whether you want CICS to perform journal resource security checking. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with a transaction is authorized to access the referenced journal. Such checking is performed every time a transaction tries to access a CICS journal.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC is active for the resource definitions.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF using the default CICS resource class name of CICSJCT prefixed by a J or K, to check whether the userid associated with a transaction is authorized to access CICS journals referenced by the transaction. The resource class name is JCICSJCT and the grouping class name is KCICSJCT.

**name** CICS calls RACF, using the specified resource class name prefixed by J or K, to verify that the userid associated with a transaction is authorized to access CICS journals.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any journal resource security checks, allowing any user to access any CICS journal.

**Restrictions** You can specify the XJCT parameter in the SIT, PARM, or SYSIN only.

**XLT={NO|xx|YES}**

specifies a suffix for the transaction list table. ( page “Defining CICS resource table and module keywords” on page 162.) The table contains a list of transactions that can be attached during the first quiesce stage of system termination.

**YES** The default transaction list table, DFHXLT, is used.

**xx** The transaction list table DFHXLTxx is used.

**NO** A transaction list table is not used.

For guidance information about coding the macros for this table, see the *CICS Resource Definition Guide*

**XPCT={YES|name|NO}**

specifies whether you want CICS to perform started transaction resource security checking, and optionally specifies the name of the RACF resource class name in which you have defined the started task security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with a transaction is authorized to use started transactions and related EXEC CICS commands. Such checking is performed every time a transaction tries to use a started transaction or one of the EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, or SET TRANSACTION.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the resource definitions.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF using the default CICS resource class name CICSPCT prefixed with A or B, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands.

The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

**name** CICS calls RACF, using the specified resource class name, to verify that the userid associated with a transaction is authorized to use started transactions or related EXEC CICS commands. The resource class name is ACICSPCT and the grouping class name is BCICSPCT.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any started task resource security checks, allowing any user to use started transactions or related EXEC CICS commands.

**Restrictions** You can specify the XPCT parameter in the SIT, PARM, or SYSIN only.

**XPPT={YES|name|NO}**

specifies that CICS is to perform application program resource security checks, and optionally specifies the RACF resource class name in which you have defined the program resource security profiles. Such checking is performed every time a transaction tries to invoke another program by using one of the CICS commands: LINK, LOAD, or XCTL.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the resource definitions.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF, using the default resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is MCICSPPT and the grouping class name is NCICSPPT.

**name** CICS calls RACF, with the specified resource class name prefixed by M or N, to verify that the userid associated with a transaction is authorized to use LINK, LOAD, or XCTL commands to invoke other programs. The resource class name is *Mname* and the grouping class name is *Nname*.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any application program authority checks, allowing any user to use LINK, LOAD, or XCTL commands to invoke other programs.

**Restrictions** You can specify the XPPT parameter in the SIT, PARM, or SYSIN only.

**XPSB={YES|name|NO}**

specifies whether you want CICS to perform program specification block (PSB) security checking, and optionally specifies the RACF resource class name in which you have defined the PSB security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to check that the userid associated with a transaction is authorized to access PSBs (which describe databases and logical message destinations used by application programs). Such checking is performed every time a transaction tries to access a PSB.

**Notes:**

1. The checking is performed only if you have specified YES for the SEC system initialization parameter and specified the RESSEC(YES) option on the resource definitions.
2. If you require security checking for PSBs to apply to remote users who access this region by means of transaction routing, the system initialization parameter PSBCHK=YES must be specified. For further information about the PSBCHK system initialization parameter, see page “PSBCHK” on page 225.

For information about preparing for and using security with CICS, see the *CICS RACF Security Guide*.

**YES** CICS calls RACF, using the default resource class name CICSPSB prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is PCICSPSB and the grouping class name is QCICSPSB.

**name** CICS calls RACF, using the specified resource class name prefixed by P or Q, to verify that the userid associated with a transaction is authorized to access PSBs. The resource class name is *Pname* and the grouping class name is *Qname*.

The resource class name specified must be 1 through 7 characters.

**NO** CICS does not perform any PSB resource security checks, allowing any user to access any PSB.

**Restrictions** You can specify the XPSB parameter in the SIT, PARM, or SYSIN only.

**XRF={NO|YES} (active and alternate)**

specifies whether XRF support is to be included in the CICS region. If the CICS region is started with the START=STANDBY system initialization parameter specified, the CICS region is the **alternate CICS region**. If the CICS region is started with the START=AUTO, START=INITIAL or START=COLD system initialization parameter specified, the CICS region is the **active CICS region**. The active CICS region signs on as such to the CICS availability manager. For background information about XRF, see the *CICS/ESA 3.3 XRF Guide*.

**XRFSOFF={NOFORCE|FORCE}**

This parameter is now obsolete and is only kept for compatibility. If it is specified, it is rejected with a message and RSTSIGNOFF is assumed.

**XRFSTME**={5|decimal-value}

This parameter is now obsolete and is only kept for compatibility. If it is specified, it is rejected with a message and RSTSIGNTIME is assumed.

**XTRAN**={YES|name|NO}

specifies whether you want CICS to perform transaction-attach security checking, and optionally specifies the RACF resource class name in which you have defined the transaction security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with the transaction is permitted to run the transaction.

#  
#

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter.

**YES** CICS calls RACF, using the default CICS resource class name of CICSTRN prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is TCICSTRN and the grouping class name is GCICSTRN.

**name** CICS calls RACF, using the specified resource class name prefixed by T or G, to verify that the userid associated with the transaction is authorized to run the transaction. The resource class name is *Tname* and the corresponding grouping class name is *Gname*.

The name specified must be 1 through 7 characters.

**NO** CICS does not perform any transaction-attach security checks, allowing any user to run any transaction.

**Restrictions:** You can specify the XTRAN parameter in the SIT, PARM, or SYSIN only.

**XTST**={YES|name|NO}

specifies whether you want CICS to perform temporary storage security checking, and optionally specifies the RACF resource class name in which you have defined the temporary storage security profiles. If you specify YES, or a RACF resource class name, CICS calls RACF to verify that the userid associated with a temporary storage request is authorized to access the referenced temporary storage queue.

**Note:** The checking is performed only if you have specified YES for the SEC system initialization parameter, specified the RESSEC option on the resource definitions, and specified TYPE=SECURITY in the temporary storage table (TST).

**YES** CICS calls RACF, using the default CICS resource class name of CICSTST prefixed by S or U, to verify that the userid associated with the transaction is authorized to access temporary storage queues referenced by the transaction. The resource class name is SCICSTST and the corresponding grouping class name is UCICSTST.

**name** CICS calls RACF, using the specified resource class name prefixed by S or U, to verify that the userid associated with a transaction is authorized to access temporary storage queues.

The name specified must be 1 through 7 characters.

**NO** CICS does not perform any temporary storage security checks, allowing any user to access any temporary storage queue.

**Restrictions** You can specify the XTST parameter in the SIT, PARM, or SYSIN only.

**XUSER={YES|NO}**

specifies whether CICS is to perform surrogate user checks.

**YES** CICS is to perform surrogate user checking in all those situations that permit such checks to be made (for example, on EXEC CICS START commands without an associated terminal). Surrogate user security checking is also performed by CICS against userids installing or modifying DB2 resource definitions that specify AUTHID or COMAUTHID.

**Note:** The XUSER parameter is also used by CICS to control access to the AUTHTYPE and COMAUTHTYPE parameters on DB2 resource definitions, although not through surrogate user checks. For more information about AUTHTYPE and COMAUTHTYPE parameters, see the *CICS Resource Definition Guide*.

For information about the various circumstances in which CICS performs surrogate user checks, see the *CICS RACF Security Guide*.

**NO** CICS is not to perform any surrogate user checking.

**Restrictions** You can specify the XUSER parameter in the SIT, PARM, or SYSIN only.

---

## The default system initialization table

The macro source statements used to assemble the default system initialization table are given in Table 25. This default SIT is in CICSTS31.CICS.SDFHAUTH, and its source, named DFHSIT\$\$, is in CICSTS31.CICS.SDFHSAMP.

Table 25. DFHSIT, the pregenerated default system initialization table

Parameter	Default value	Description
ADI	30	XRF(B) - Alternate delay interval
AICONS	NO	No autoinstall for MVS CONSOLES
AIEXIT	DFHZATDX	Autoinstall user program name
AILDELAY	0	Delete delay period for AI terminals
AIQMAX	100	Maximum no. of terminals queued for AI
AIRDELAY	700	Restart delay period for AI terminals
AKPFREQ	4000	Activity keypoint frequency
APPLID	DBDCCICS	VTAM APPL identifier
AUTCONN	0	Autoconnect delay
AUTODST	NO	Lang env automatic storage tuning
AUTORESETTIME	NO	Time-of-day synchronization
AUXTR	OFF	Auxiliary trace option

#

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
AUXTRSW	NO	Auxiliary trace autoswitch facility
BMS	(FULL,,UNALIGN,DDS)	Basic Mapping Support options
CICSSVC	216	The CICS SVC number
CILOCK	NO	Don't keep CI lock after read update
CLSDSTP	NOTIFY	Notification for ISSUE PASS command
CLT		The command list table option/suffix
CMDPROT	YES	Exec storage command checking
CMDSEC	ASIS	API command security checking
CONFDATA	SHOW	Show confidential data in dump/trace
CONFTXT	NO	Don't prevent VTAM tracing user data
CRLPROFILE		Name of profile that allows CICS to access certificate revocation lists
CSDACC	READWRITE	CSD access
CSDBKUP	STATIC	Backup type of CSD (STATIC or DYNAMIC)
CSDBUFND		Number of data buffers for the CSD
CSDBUFNI		Number of index buffers for the CSD
CSDDISP		CSD Disposition for dynamic allocation
CSDDSN		CSD data set name for dynamic allocation
CSDFRLOG	NO	Journal id. for CSD forward recovery
CSDINTEG	UNCOMMITTED	Read integrity = uncommitted
CSDJID	NO	Journal id. for CSD auto. journaling
CSDLRNO	1	The VSAM LSR pool number for the CSD
CSDRECOV	NONE	CSD recoverable file option
CSDRLS	NO	Use traditional VSAM
CSDSTRNO	2	CSD Number of strings
CWAKEY	USER	CWA storage key
DAE	NO	SDUMPS will not be suppressed by DAE
DATFORM	MMDDYY	CSA date format
DB2CONN	NO	Do not connect to DB2 at CICS startup
DBCTLCON	NO	Do not connect to DBCTL at CICS start
DFLTUSER	CICSUSER	Default user
DIP	NO	Batch data interchange program
DISMACP	YES	Disable macro programs
DOCCODEPAGE	037	Default host code page
DSALIM	5M	Upper limit of DSA below 16Mb line
DSHIPIDL	020000	Delete shipped idle time
DSHIPINT	120000	Delete shipped interval
DSRTPGM	NONE	Distributed routing program
DTRPGM	DFHDYP	Dynamic routing program
DTRTRAN	CRTX	Default dynamic tran routing transid

#

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
DUMP	YES	Dump option
DUMPDS	AUTO	CICS dump data set opening option
DUMPSW	NO	Dump data set autoswitch option
DURETRY	30	SDUMP total retry time (in seconds)
EDSALIM	30M	Upper limit of DSA above 16MB line
EJBROLEPRFX		EJB ROLE PREFIX
ENCRYPTION	STRONG	Level of encryption for SSL
EODI	E0	End-of-data indicator for sequential devices
ESMEXITS	NOINSTLN	External security manager exits
FCT	NO	File control table option/suffix
FEPI	NO	Front-End Programming Interface
FLDSEP		End-of-field separator characters
FLDSTRT		Field start character for built in function
FORCEQR	NO	Don't force QR for threadsafe programs
FSSTAFF	NO	Function-shipped START affinity option
FTIMEOUT	30	File timeout 30 seconds
GMTEXT	'WELCOME TO CICS'	Good-morning message text
GMTRAN	CSGM	Initial transaction
GTRAN	NO	Signoff transaction
GRNAME		Generic resource name for CICS TORs
GRPLIST	DFHLIST	List name of CSD groups for startup
GTFTR	OFF	GTF trace option
HPO	NO	VTAM High Performance Option (HPO)
ICP		Interval control program start option
ICV	1000	Region exit interval (milliseconds)
ICVR	5000	Runaway task interval (milliseconds)
ICVTSD	500	Terminal scan delay interval ( " )
INITPARM		Initialization parms for programs
INTTR	ON	CICS internal trace option
IRCSTRT	NO	Interregion communication start
ISC	NO	Intersystem communication option
JESDI	30	JES delay interval for XRF alternate
JVMCCPROFILE	DFHJVMCC	Master JVM SharedClassCache profile
JVMCCSIZE	24M	Master JVM init SharedClassCache size
JVMCCSTART	AUTO	Start SharedClassCache when needed
JVMPROFILEDIR	/usr/lpp/cicsts /cicsts31/JVMProfiles	JVM profile directory
KEYRING		Key ring to be used by SSL support
LGDFINT	30	Log defer interval in Log Manager
LGNMSG	NO	Extract VTAM logon data

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
LLACOPY	YES	Use MVS LLACOPY support
LPA	NO	Use-LPA option for CICS/user modules
MAXJVMTCBS	5	Maximum number of JVM open TCBs
MAXOPENTCBS	12	Maximum number of open TCBs
MAXSOCKETS	65535	Maximum number of IP sockets
MAXSSLTCBS	8	Limit on number of SSL TCBs
MAXXPTCBS	5	Limit on number of XP TCBs
MCT	NO	Monitoring control table option/suffix
MN	OFF	CICS monitoring option
MNCONV	NO	Monitoring converse recording option
MNEVE	OFF	Monitoring event class option
MNEXC	OFF	Monitoring exception class option
MNFREQ	0	Monitoring frequency period
MNPER	OFF	Monitoring performance class option
MNRES	OFF	Monitoring resource class option
MNSUBSYS		Monitoring subsystem identification
MNSYNC	NO	Monitoring syncpoint recording option
MNTIME	GMT	Monitoring timestamp (GMT/LOCAL)
MQCONN	NO	Do not connect to MQ at startup
MROBTCH	1	Number of MRO requests to batch
MROFSE	NO	Extend lifetime of Long-running mirror
MROLRM	NO	Long-running mirror task option
MSGCASE	MIXED	CICS messages in mixed case
MSGLVL	1	System console MSG level option
MXT	5	Maximum number of tasks in CICS
NATLANG	E	List of national languages
NCPLDFT	DFHNC001	Named counter default pool name
OPERTIM	120	Write to operator timeout (seconds)
OPNDLIM	10	OPNDST/CLSDST request limit
PARMERR	INTERACT	System init. parameter errors option
PDI	30	Primary delay interval - XRF active
PDIR	NO	DL/I PSB directory option/suffix
PGAICTLG	MODIFY	PG autoinstall catalog state
PGAIXIT	DFHPGADX	PG autoinstall exit program
PGAIPGM	INACTIVE	PG autoinstall state
PGCHAIN		BMS CHAIN command
PGCOPY		BMS COPY command
PGPURGE		BMS PURGE command
PGRET		BMS RETURN command
PLTPI		Program list table PI option/suffix

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
PLTPISEC	NONE	No PLT security checks on PI programs
PLTPIUSR		PLT PI userid = CICS region userid
PLTSD	NO	Program list table SD option/suffix
PRGDLAY	0	BMS purge delay interval
PRINT	NO	Print key option
PRTYAGE	32768	Dispatcher priority ageing value
PSBCHK	NO	PSB resource checking required
PSDINT	0	Persistent session delay interval
PSTYPE	SNPS	VTAM single node persistent Sessions
PVDELAY	30	Timeout value for LUIT table
QUIESTIM	240	Timeout value for quiesce requests
RAMAX	256	Maximum I/O area for RECEIVE ANY
RAPOOL	50	Maximum RECEIVE ANY request parm. lists
RENTPGM	PROTECT	Reentrant program write protection
RESP	FME	Logical unit response type
RESSEC	ASIS	Resource security check
RLS	NO	RLS option
RLSTOLSR	NO	RLS files in LSRPOOL build calculation
RMTRAN	CSGM	XRF alternate recovery transaction
RRMS	NO	Recoverable resource management services
RST	NO	Recovery service table (XRF-DBCTL)
RSTSIGNOFF	NOFORCE	XRF - Re-sign on after takeover
RSTSIGTIME	5	XRF - sign off timeout value
RUWAPool	NO	Allocating storage pool for Language Environment
SDTRAN	CESD	Shutdown transaction
SEC	YES	External security manager option
SECPRFX	NO	Security prefix
SKRPA1		SKR PA1 PAGE RETRIEVAL CMD
SKRPA2		SKR PA2 PAGE RETRIEVAL CMD
SKRPA3		SKR PA3 PAGE RETRIEVAL CMD
SKRPF1		SKR PF1 PAGE RETRIEVAL CMD
SKRPF2		SKR PF2 PAGE RETRIEVAL CMD
SKRPF3		SKR PF3 PAGE RETRIEVAL CMD
SKRPF4		SKR PF4 PAGE RETRIEVAL CMD
SKRPF5		SKR PF5 PAGE RETRIEVAL CMD
SKRPF6		SKR PF6 PAGE RETRIEVAL CMD
SKRPF7		SKR PF7 PAGE RETRIEVAL CMD
SKRPF8		SKR PF8 PAGE RETRIEVAL CMD
SKRPF9		SKR PF9 PAGE RETRIEVAL CMD

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
SKRPF10		SKR PF10 PAGE RETRIEVAL CMD
SKRPF11		SKR PF11 PAGE RETRIEVAL CMD
SKRPF12		SKR PF12 PAGE RETRIEVAL CMD
SKRPF13		SKR PF13 PAGE RETRIEVAL CMD
SKRPF14		SKR PF14 PAGE RETRIEVAL CMD
SKRPF15		SKR PF15 PAGE RETRIEVAL CMD
SKRPF16		SKR PF16 PAGE RETRIEVAL CMD
SKRPF17		SKR PF17 PAGE RETRIEVAL CMD
SKRPF18		SKR PF18 PAGE RETRIEVAL CMD
SKRPF19		SKR PF19 PAGE RETRIEVAL CMD
SKRPF20		SKR PF20 PAGE RETRIEVAL CMD
SKRPF21		SKR PF21 PAGE RETRIEVAL CMD
SKRPF22		SKR PF22 PAGE RETRIEVAL CMD
SKRPF23		SKR PF23 PAGE RETRIEVAL CMD
SKRPF24		SKR PF24 PAGE RETRIEVAL CMD
SKRPF25		SKR PF25 PAGE RETRIEVAL CMD
SKRPF26		SKR PF26 PAGE RETRIEVAL CMD
SKRPF27		SKR PF27 PAGE RETRIEVAL CMD
SKRPF28		SKR PF28 PAGE RETRIEVAL CMD
SKRPF29		SKR PF29 PAGE RETRIEVAL CMD
SKRPF30		SKR PF30 PAGE RETRIEVAL CMD
SKRPF31		SKR PF31 PAGE RETRIEVAL CMD
SKRPF32		SKR PF32 PAGE RETRIEVAL CMD
SKRPF33		SKR PF33 PAGE RETRIEVAL CMD
SKRPF34		SKR PF34 PAGE RETRIEVAL CMD
SKRPF35		SKR PF35 PAGE RETRIEVAL CMD
SKRPF36		SKR PF36 PAGE RETRIEVAL CMD
SNSCOPE	NONE	Multiple CICS sessions per userid
SPCTR	(1,2)	Level(s) of special tracing required
SPOOL	NO	System spooling interface option
SRBSVC	215	HPO Type 6 SVC number
SRT	1\$	System recovery table option/suffix
SSLCACHE	CICS	SSL session id caching
SSLDELAY	(600,number)	SSL timeout value
START	AUTO	CICS system initialization option
STARTER	YES	Starter (\$ and &numsign;) suffixes option
STATRCD	OFF	statistics recording status
STGPROT	NO	Storage protection facility
STGRCVY	NO	Storage recovery option
STNTR	1	Level of standard tracing required

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
SUBTSKS	0	Number of concurrent mode TCBs
SUFFIX	\$\$	Suffix of this SIT
SYDUMAX	999	No of SYSDUMPS to be taken
SYSIDNT	CICS	Local system identifier
SYSTR	ON	Master system trace flag
TAKEOVR	MANUAL	XRF alternate takeover option
TBEXITS		Backout exit programs
TCP	YES	Terminal control program option/suffix
TCPIP	NO	TCP/IP support
TCSACTN	NONE	TC Shutdown action
TCSWAIT	4	TC Shutdown wait
TCT	NO	Terminal control table option/suffix
TCTUAKEY	USER	TCT user area storage key
TCTUALOC	BELOW	TCT user area below 16MB
TD	(3,3)	Transient data buffers and strings
TDINTRA	NOEMPTY	Initial state of transient data queues
TDSUBTASK	OFF	No TD subtasking
TRANISO	NO	Transaction Isolation
TRAP	OFF	F.E. global trap exit option
TRDUMAX	999	No of TRANDUMPS to be taken
TRTABSZ	16	Internal trace table size in 1K bytes
TRTRANSZ	16	Transaction dump trace table size
TRTRANTY	TRAN	Transaction dump trace option
TS	(3,3)	Temporary storage buffers and strings
TST	NO	Temporary storage table option/suffix
UOWNETQL		Qualifier for NETUOWID
USERTR	ON	Master user trace flag
USRDELAY	30	Timeout value for User Dir. Entries
VTAM	YES	VTAM access method option
VTPREFIX	\	Client virtual terminal prefix
WEBDELAY	(5,60)	Web timer values
WRKAREA	512	Common work area (CWA) size in bytes
XAPPC	NO	RACF class APPCLU required
XCMD	YES	SPI use default name for RACF check
XDCT	YES	DCT use default name for RACF check
XDB2	NO	No RACF security profile for DB2
XEJB	YES	EJB security required
XFCT	YES	FCT use default name for RACF check
XJCT	YES	JCT use default name for RACF check
XLT	NO	Transaction list table option/suffix

#

Table 25. DFHSIT, the pregenerated default system initialization table (continued)

Parameter	Default value	Description
XPCT	YES	PCT use default name for RACF check
XPPT	YES	PPT use default name for RACF check
XPSB	YES	PSB use default name for RACF check
XRF	NO	Extended recovery feature (XRF) option
XTRAN	YES	Transid use default name, RACF check
XTST	YES	TST use default name for RACF check
XUSER	YES	Surrogate user checking to be done

---

## Assembling the SIT

When you have coded the DFHSIT macro, you must assemble and link-edit the table into an APF-authorized library, such as CICSTS31.CICS.SDFHAUTH, and include this library in the STEPLIB concatenation of the startup job stream. For information about assembling and link-editing CICS control tables, and an explanation of the syntax notation used to describe CICS macros, see the *CICS Resource Definition Guide*.

## Assembler errors from undefined keywords

If you code a system initialization parameter in your SIT source, and the parameter's keyword is not defined in the CICS-supplied version of the DFHSIT macro, you get an IEV017 warning message from the assembly, as follows:

```
IEV017 ** WARNING ** UNDEFINED KEYWORD PARAM. DEFAULT TO
          POSITIONAL, INCLUDING KW -- OPENC/aaaaaaa
```

where: aaaaaaa is the unidentified keyword.

However, be aware that because of work space limitations there is a limit to the number of undefined keyword errors that the assembler can generate. This means that if your SIT contains more undefined keywords than the assembler can generate messages for, some are not flagged until a second (or even later) assembly, and as you correct flagged errors, other errors (previously unflagged) may appear during reassembly.

---

## Selecting versions of CICS programs and tables

A CICS program is usually made up from a group of related CICS functional modules, one example of which is the terminal control program. For most CICS programs you can only have one version, which is supplied with CICS. However, for some CICS programs you can create more than one version; for example, with different service levels. To select a particular version of a program, you can include the load library containing that version in the CICS startup JCL. For the basic mapping support (BMS) suite, however, you can select from different versions, by explicitly selecting the level of function needed.

You can also specify that a program is **not** needed (see "Excluding unwanted programs" on page 271 for details).

You can use these methods **only** for the programs referred to in this section and in "Excluding unwanted programs" on page 271, by coding system initialization parameters.

## Using an explicit level of function to select programs

You use an explicit level of function to select the BMS suite of programs. When you specify your BMS requirement on the system initialization parameter BMS, you can select one of three versions. The BMS level of function is selected by the parameter options MINIMUM, STANDARD, or FULL, from which the system initialization program loads the set of programs you require.

## Excluding unwanted programs

The three ways of excluding programs that are not required are by specifying:

1. programname=NO
2. tablename=NO
3. function=NO

### Specifying programname=NO

If you code *programnamesystem* initialization parameter=NO as a , (for example, DIP=NO), you exclude the named management program at CICS system initialization.

The programs that you can exclude by coding *programname:=NO* are

- Batch data interchange program (DIP)
- Terminal control program (TCP)

**Note:** In the case of DIP, you get a dummy version of the management program, which is supplied on the distribution tape with a suffix of **DY**.

### Specifying tablename=NO for the programs control table

Not all of the CICS programs have a programname parameter in the SIT. Ansystem initialization parameter alternative method is to code NO on the for the associated table. This has the same effect as coding NO against a program name parameter, and the associated CICS program is excluded at system initialization, either by loading a dummy program, or by some other technique.

The system recovery table (SRT) can be used in this way, and the associated system recovery program (SRP) will be excluded.

### The dummy TCT, DFHTCTDY

There is a special case where you can also specify tablename=NO, but this does not load a dummy terminal control program. You specify TCT=NO when you are using resource definition online, and all your terminal resource definitions are in the CSD.

When you specify TCT=NO, CICS loads a dummy TCT named DFHTCTDY. A pregeneratedCICSTS31.CICS dummy table of this name is supplied in .SDFHLOAD, and the sourceCICSTS31.CICS statements of DFHTCTDY are supplied in .SDFHSAMP. If you specify TCT=NO, a generated table of this name must be available in a library of the DFHRPL concatenation when you start CICS.

The dummy TCT provides **only** the CICS and VTAM control blocks that you need if you are using VTAM terminals and using the CSD for storing terminal definitions. You define your VTAM terminals using the RDO transaction, CEDA, or the DEFINE command of the CSD batch utility program, DFHCSDUP.

### **Specifying function=NO**

If you code function=NO as a system initialization parameter (for example, XRF=NO), you exclude the management program associated with the named function at CICS system initialization.

You can exclude intersystem communication (ISC), the 3270 print-request facility, the system spooling interface, or the extended recovery facility (XRF), in this way.

---

## Chapter 19. Processing system initialization parameters

This chapter describes the CICS system initialization process as follows:

1. A brief introduction to the process of how you supply system initialization parameters, and the role of the CICS parameter manager domain in this process
2. An explanation of how CICS uses the special system initialization keywords
3. A description of the start and restart classes, and how they are controlled

This chapter describes:

- “Supplying system initialization parameters to CICS”
- “Using system initialization control keywords”
- “Controlling start and restart” on page 278

---

### Supplying system initialization parameters to CICS

The CICS parameter manager domain loads a system initialization table (SIT) at the start of the initialization process. You specify the SIT that defines the CICS characteristics appropriate to your needs by coding the suffix of the DFHSITxx load module (where xx is the suffix) on the SIT= system initialization parameter. If you fail to specify the suffix of a SIT, then CICS tries to load an unsuffixed module.

You can modify many of the system initialization parameters dynamically at the beginning of CICS initialization by providing system initialization parameters in the startup job stream, or through the system console. There are also some system initialization parameters that you cannot code in the SIT, and can only supply at startup time. You specify system initialization parameters at startup time in any of three ways:

1. In the PARM parameter of the EXEC PGM=DFHSIP statement
2. In the SYSIN data set defined in the startup job stream
3. Through the system operator’s console

You can use just one of these methods, or two, or all three. However, CICS processes these three sources of input in strict sequence, as follows:

1. The PARM parameter
2. The SYSIN data set (but only if SYSIN is coded in the PARM parameter; see “SYSIN” on page 274)
3. The console (but only if CONSOLE is coded in either the PARM parameter or in the SYSIN data set; see “CONSOLE(CN)” on page 274)

**Note:** If you supply duplicate system initialization parameters, either through the same or a different medium, CICS takes the last one that it reads. For example, if you specify MCT=1\$ in the PARM parameter, MCT=2\$ in the SYSIN data set, and finally enter MCT=3\$ through the console, CICS loads DFHMCT3\$.

---

### Using system initialization control keywords

You can use the following control keywords at startup to control how CICS is to read system initialization parameters:

1. SYSIN (or SI for short)
2. CONSOLE (or CN for short)
3. .END

The purpose of these special keywords, and where you can code them, are described as follows:

### **SYSIN (SI)**

This keyword tells CICS to read initialization parameters from the SYSIN data set.

**Where to code SYSIN:** You can code SYSIN (or SI) only in the PARM parameter of the EXEC PGM=DFHSP statement. The keyword can appear once only and must be at the end of the PARM parameter. CICS does not read SYSIN until it has finished scanning all of the PARM parameter, or until it reaches a .END before the end of the PARM parameter. (See .END on page "END.")

#### **Examples:**

```
//stepname EXEC PGM=DFHSP,PARM='SIT=6$,SYSIN,.END'  
//stepname EXEC PGM=DFHSP,PARM='SIT=6$,DLI=YES,SYSIN,.END'
```

### **CONSOLE (CN)**

This keyword tells CICS to read initialization parameters from the console. CICS prompts you with message DFHPA1104 when it is ready to read parameters from the console.

**Where to code CONSOLE:** You can code CONSOLE (or CN) in the PARM parameter of the EXEC PGM=DFHSP statement or in the SYSIN data set. This keyword can appear either at the end of the PARM parameter or in the SYSIN data set, but code it in one place only.

If you code CONSOLE (or CN) in the PARM parameter, and PARM also contains the SYSIN keyword, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIN data set. Similarly, wherever you place the CONSOLE keyword in the SYSIN data set, CICS does not begin reading parameters from the console until it has finished reading and processing the SYSIN data set.

#### **Examples:**

```
//stepname EXEC PGM=DFHSP,PARM='SIT6$,CONSOLE,.END'  
//stepname EXEC PGM=DFHSP,PARM='CONSOLE,SYSIN,.END'  
//stepname EXEC PGM=DFHSP,PARM='SIT=6$,CN,SI,.END'
```

#### **Note:**

If both SYSIN (or SI) and CONSOLE (or CN) appear as keywords of the PARM parameter, the order in which they are coded is irrelevant as long as no other keywords, other than .END, follow them.

### **.END**

The meaning of this keyword varies:

**PARM** The use of the .END keyword is optional in the PARM parameter. If you omit it, CICS assumes it to be at the end of the PARM parameter. If you code .END in the PARM parameter it can have one of two meanings:

1. If you also code one, or both, of the other control keywords (CONSOLE and/or SYSIN) .END denotes the end of the PARM parameter only.

For example:

```
//stepname EXEC PGM=DFHSP,PARM='SIT6$,SI,CN,.END'
```

2. If you code .END as the only control keyword in the PARM parameter, it denotes the end of all system initialization parameters, and CICS begins the initialization process. For example:

```
//stepname EXEC PGM=DFHSIP,PARM='SIT=6$, .END'
```

If .END is not the last entry in the PARM parameter, CICS truncates the PARM parameter and the parameters following the .END keyword are lost.

**SYSIN** The use of the .END keyword is optional in the SYSIN data set. If you omit it, CICS assumes it to be at the end of SYSIN. If you code .END in the SYSIN data set its meaning depends on your use of the CONSOLE keyword, as follows:

- If you code the CONSOLE control keyword in the PARM parameter or in the SYSIN data set, .END denotes the end of the SYSIN data set only.
- If you do not code the CONSOLE control keyword in the PARM parameter or in the SYSIN data set, .END denotes the end of all CICS system initialization parameters, and CICS begins the initialization process.

If you code .END, and it is not the last entry in the SYSIN data set, or not at the end of a SYSIN record, CICS initialization parameters following the .END are ignored. To avoid accidental loss of initialization parameters, ensure that the .END keyword is on the last record in the SYSIN data set, and that it is the only entry on that line. (However, if you want to remove some system initialization parameters from a particular run of CICS, you could position them after the .END statement just for that run.)

The following example shows the use of .END in a SYSIN data set:

```
//SYSIN DD *
* CICS system initialization parameters
SIT=6$,START=COLD,
* XRF=NO,          ( XRF this run - SIT defines XRF=YES
PDIR=1$,          ( SUFFIX of PSB directory
:
:
.END
/*
```

## CONSOLE

The meaning of .END through the console depends on whether you are entering new parameters or entering corrections. The two meanings are as follows:

1. If you are keying new parameters in response to message DFHPA1104, .END terminates parameter reading, and CICS starts initialization according to the SIT it has loaded, but modified by any system initialization parameters you have supplied. Until you enter the .END control keyword, CICS continues to prompt you for system initialization parameters.
2. If you have coded PARMERR=INTERACT, and CICS detects a parameter error, either in the keyword or in the value that you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915. If you enter the correct keyword or value, initialization resumes with CICS continuing to process either the PARM parameter, the SYSIN data set, or prompting for more system initialization parameters through the console, depending on where CICS detected the error. If you cannot correct the error, but want CICS to continue with the initialization process, you can enter .END to end the error correction phase.

## Processing the PARM parameter

If you omit the PARM parameter from the EXEC PGM=DFHSIP statement, CICS assumes that there are no SIT override or other initialization parameters, and tries to load an unsuffixed module named DFHSIT. As a general rule, this is unlikely to be your intention, so the PARM parameter should at least specify the suffix of your system initialization table, using the SIT keyword. Alternatively, you can code the special SYSIN keyword as the only PARM parameter, and supply the suffix of your SIT and the other system initialization parameters from the SYSIN data set.

CICS scans the PARM string looking for a SIT= parameter, any of the special control keywords, or any system initialization parameters, and proceeds as follows:

- If CICS finds a SIT= parameter but no SYSIN keyword, CICS tries to load the SIT as soon as it has finished scanning the PARM parameter. Processing any CICS system initialization parameters that are also present in the PARM parameter takes place only after the SIT has been loaded.
- If CICS finds a SIT= parameter and also a SYSIN keyword, CICS does not try to load the SIT until it has also finished scanning the SYSIN data set. In this case, loading the SIT is deferred because there can be other SIT= parameters coded in the SYSIN data set that override the one in the PARM parameter.

Processing any system initialization parameters that are also present in the PARM parameter takes place only after the SIT has been loaded.

### Rules for coding PARM parameters

The rules for coding the PARM parameter on an EXEC job control statement are described fully in the *OS/390 MVS JCL Reference* manual. Briefly, the maximum number of characters you can code is 100, excluding the opening and closing delimiters, which can be either apostrophes or parentheses. All CICS system initialization parameters must be separated by a comma, and the separating commas are included in the 100 character limit. Because of this limiting factor, you might prefer to limit the use of the PARM parameter to specify the SYSIN control keyword only.

## Processing the SYSIN data set

CICS scans the SYSIN data set looking for a SIT= parameter and any of the special keywords, as well as system initialization parameters.

If CICS finds a SIT= parameter in SYSIN, it tries to load that SIT, overriding any that was specified in the PARM parameter. If CICS does not find a SIT= parameter in SYSIN, it tries to load any SIT specified in the PARM parameter.

However, if after scanning the PARM parameter and the SYSIN data set CICS has not found a SIT= parameter, CICS does one of the following:

1. If you specified CONSOLE in the PARM parameter or in the SYSIN data set, CICS prompts you with the following message to enter the SIT suffix as the first parameter through the console:

```
rr DFHPA1921 DBDCCICS PLEASE SPECIFY THE REQUIRED SIT SUFFIX, OR
                    SPECIFY 'NONE' (UNSUFFIXED).
```

2. If you did not specify CONSOLE, CICS tries automatically to load an unsuffixed SIT module (DFHSIT). If this load fails, CICS issues message DFHPA1106, requesting a SIT suffix in reply to the message.

**Note:** CICS does not process any system initialization parameters that are coded in the PARM parameter and the SYSIN data set until after the SIT has been loaded.

## Rules for coding CICS system initialization parameters in the SYSIN data set

There are a few rules to observe when coding CICS system initialization parameters in the SYSIN data set. These are:

- You must use a comma to separate parameters that are on the same line, but the use of a comma at the end of a SYSIN record is optional.
- You can use an asterisk in column 1 to code comments, or to remove temporarily an initialization parameter from a particular execution of CICS.
- You can also add comments after the parameters on a SYSIN line, but they must be preceded by at least one blank character.
- SYSIN is an 80-byte file, and everything that appears in positions 1 through 80 is treated by CICS as input data.
- You can continue, on another line in SYSIN, parameters that have multiple operands if you make the split immediately after a comma. CICS concatenates the operands, omitting the intervening blanks.
- As a general rule, you cannot split an individual operand between lines. However, in the case of the GMTEXT parameter, you can enter the operand over more than one line up to the maximum of 246 characters. The format of this parameter is:

```
GMTEXT='User' 's text'
```

You can use apostrophes to punctuate message text, provided that you code *two* successive apostrophes to represent a single apostrophe (as shown in the example above). The apostrophes delimiting the text are mandatory.

- You must take care when coding parameters that use apostrophes, parentheses, or commas as delimiters, because failure to include the correct delimiters is likely to cause unpredictable results.

## Processing the console entries

Generally, CICS does not begin to read from the console until it has loaded the SIT and processed any initialization parameters that are coded in the PARM parameter and the SYSIN data set. CICS accepts system initialization parameters from the console until you terminate the input with `.END`.

You can specify a `SIT=` parameter only as the first parameter through the console when prompted by message DFHPA1921, at which point CICS tries to load the specified SIT. If you try to specify a `SIT=` parameter after CICS has loaded the SIT it is rejected as an error.

## Rules for coding CICS system initialization parameters at the console

When it is ready to read parameters from the console, CICS displays the following message (where `nn` is the reply ID):

```
nn DFHPA1104 applid - SPECIFY ALTERNATIVE SIT PARAMETERS, IF ANY,  
AND THEN TYPE '.END'.
```

You can enter as many initialization parameters as you can get on one line of the console, but you must use a comma to separate parameters. CICS continues to prompt for system initialization parameters with displays of message DFHPA1105 until you terminate console input by entering the `.END` control keyword.

## Entering corrections to initialization parameters through the console

If you have coded PARMERR=INTERACT, and CICS detects a parameter error, either in the keyword or in the value you have assigned to it, CICS prompts you to correct the error with message DFHPA1912 or DFHPA1915:

```
DFHPA1912 'applid' SIT OVERRIDE 'keyword' IS NOT RECOGNIZED.  
          SPECIFY CORRECT SIT OVERRIDE.  
DFHPA1915 'applid' INVALID DATA HAS BEEN DETECTED FOR SIT OVERRIDE 'keyword'.  
          RESPECIFY THE OVERRIDE.
```

CICS prompts you to enter corrections to any errors it finds in the PARM parameter or the SYSIN data set **after** it has loaded the SIT, and as each error is detected. This means that if there is an APPLID parameter **following** the parameter that is in error, either in the PARM parameter or in the SYSIN data set, it is the APPLID coded in the SIT that CICS displays in messages DFHPA1912 and DFHPA1915.

---

## Controlling start and restart

The type of initialization that CICS performs is not only determined by the START parameter. The CICS local and global catalogs also play a major role in the initialization process, together with any system initialization parameters that you provide, either in the SIT or at run time by one of the three methods described in “Using system initialization control keywords” on page 273..

## The role of the CICS catalogs

CICS uses its catalogs to save information between CICS shutdown and the next restart.

### The global catalog

CICS uses the global catalog to save all resource definitions that are installed at CICS shutdown. These are:

- BMS maps sets and partition sets
- Connections and sessions
- Files
- Programs
- Terminals and typeterms
- Transactions and transaction profiles
- Transient data queues

The resource definitions that CICS saves at its shutdown may have been installed during a cold start (from a list of groups specified by a group list system initialization parameter), or during CICS execution (by RDO commands).

If you run CICS with START=AUTO, and a warm or emergency restart results, CICS restores all the installed resource definitions as they were at normal CICS shutdown, or at the time of system failure. The general rule is that you cannot alter installed resource definitions during a restart except by coding START=COLD or START=INITIAL. For details of the results of the possible combinations of CICS restart-type and global catalog state, see “The START system initialization parameter” on page 279.

The CICS domains also use the global catalog to save their domain status between runs. In some cases this information can be overridden during a restart by supplying system initialization parameters. For example, CICS monitoring uses the cataloged status at a restart, but modified by any system initialization parameter. In other cases the domain information saved in the catalog is always used in a restart.

For example, CICS statistics interval time is always restored from the catalog in a warm or emergency restart, because the statistics domain does not have this as a system initialization parameter. To change this you must use CEMT or EXEC CICS commands after control is given to CICS. Alternatively, you can enforce system defaults by performing a cold start.

**Note:** If you need to reinitialize the global catalog for any reason, you must also reinitialize the local catalog.

**The local catalog:** The CICS domains use the local catalog to save some of their information between CICS runs. If you delete and redefine the local catalog, you must:

- Initialize the local catalog with an initial set of domain records.
- Use the CICS-supplied utility program, DFHSMUTL, to re-add records to enable the CICS self-tuning mechanism for storage manager domain subpools. For details of how to do this see the *CICS Operations and Utilities Guide*.
- Delete and reinitialize the global catalog.

For more information about initializing the local catalog, see “Defining the local catalog” on page 91. Some of the information that is saved in the local catalog can be overridden at CICS system initialization by system initialization parameters, such as CICS transaction dump data set status.

**Note:** If you need to reinitialize the local catalog for any reason, you must also reinitialize the global catalog.

## The START system initialization parameter

You can influence the type of startup that CICS performs, by specifying the START system initialization parameter, as follows:

### START=AUTO

If you code AUTO as the START operand, CICS determines which of four possible types of start to perform by looking for two records which may or may not be present in the global catalog:

- The recovery manager control record
- The recovery manager autostart override record

Depending on whether either or both of these records exist, and their contents, CICS decides which type of start to perform:

#### 1. Initial start

CICS performs an initial start in each of the following cases:

- There is a recovery manager autostart override record that specifies AUTOINIT in the global catalog.
- The control record specifies an initial start. (This can happen if a previous initial start failed.)

If you set CICS to perform an initial start, you should reinitialize the local catalog before bringing up CICS.

#### 2. Cold start

CICS performs a cold start in the following cases:

- The recovery manager control record specifies a cold start. (This can happen if a previous cold start did not complete.)
- There is both a recovery manager control record (which specifies anything other than an initial start) *and* an autostart override record that specifies AUTOCOLD.

Log records for local resources are purged and resource definitions rebuilt from the CSD or CICS control tables. Units of work on other systems are resynchronized with this system, as described under START=COLD.

### 3. Warm start

If the recovery manager control record indicates that the previous run of CICS terminated normally with a successful warm keypoint, CICS performs a warm restart—*unless* the autostart override record specifies AUTOINIT or AUTOCOLD, in which case an initial or cold start is performed.

For the warm restart to be successful, the local catalog must contain the information saved by the CICS domains during the previous execution.

A warm start restores CICS to the state it was in at the previous shutdown.

You can modify a warm restart by coding the NEWSIT system initialization parameter. This has the effect of enforcing the system initialization parameters coded in the SIT, overriding any cataloged status from the previous CICS shutdown.

The exceptions to this is the system initialization parameter FCT, the CSDxxxxx group (for example CSDACC), and GRPLIST, which are always ignored in a warm restart, even if you specify NEWSIT=YES. Specifying NEWSIT=YES causes, in effect, a partial cold start.

### 4. Emergency start

If the control record in the global catalog indicates that the previous run of CICS terminated in an immediate or uncontrolled shutdown, CICS performs an emergency restart.

START=AUTO should be the normal mode of operation, with the choice of start being made by CICS automatically. Use the recovery manager utility program, DFHRMUTL, to set overrides.

### START=INITIAL

CICS initializes using the resource definitions specified by the system initialization parameters, ignoring any previously installed resource definitions saved in a warm keypoint in the global catalog. This includes all the groups of resources specified by the GRPLIST= system initialization parameter, and those resources specified in CICS control tables.

**Note:** The global catalog and system log are initialized, and all information in them is lost. Because recovery information for remote systems is not preserved, damage may be done to distributed units of work

You should rarely need to specify START=INITIAL; if you simply want to reinstall definitions of local resources from the CSD, use START=COLD instead.

Examples of times when an initial start is necessary are:

- When bringing up a new CICS system for the first time.
- After a serious software failure, when the system log has been corrupted.
- If the global catalog is cleared or reinitialized.
- When you want to run CICS with a dummy system log. (If the system log is defined as a dummy, it is ignored.)

If it *is* necessary to perform an initial start of CICS, you can do so in two ways:

- By specifying START=INITIAL.
- By using the recovery manager utility program, DFHRMUTL, to set the autostart override record to AUTOINIT, and specifying START=AUTO. For information about DFHRMUTL, see the *CICS Operations and Utilities Guide*.

#### START=COLD

CICS initializes using the resource definitions specified by the system initialization parameters, ignoring any previously installed resource definitions saved in a warm keypoint in the global catalog. This includes all the groups of resources specified by the GRPLIST= system initialization parameter, and those resources specified in CICS control tables.

Recovery information relating to remote systems or to RMI-connected resource managers is preserved. The CICS log is scanned during startup, and any information regarding unit of work obligations to remote systems, or to non-CICS resource managers (such as DB2) connected through the RMI, is preserved. (That is, any decisions about the outcome of local UOWs, needed to allow remote systems or RMI resource managers to resynchronize their resources, are preserved.)

Note that, on a cold start, the following are *not* preserved:

- Updates to *local* resources that were not fully committed or backed out during the previous execution of CICS. In particular, although remote systems may resynchronize their units of work successfully, local resources updated in those distributed units of work are not locked and need not be in either a committed or a backed-out state.
- Recovery information for remote systems connected by LU6.1 links, or for earlier releases of CICS systems connected by MRO.

A start initiated by START=COLD is not entirely without reference to the previous run of a CICS system using the same global catalog. If you want to perform a fully cold start of CICS, without reference to any previous execution, code START=INITIAL. If you simply want to reinstall definitions of local resources from the CSD, use START=COLD.

There may be times when it is necessary to perform a cold start of CICS, irrespective of the type of system termination that CICS recorded in the global catalog. You can do this in two ways:

- By specifying START=COLD.
- By using DFHRMUTL to set the autostart override record to AUTOCOLD, and specifying START=AUTO.

#### START=STANDBY

The STANDBY option is for use only with XRF=YES. START=STANDBY causes the alternate CICS region to initialize only to the point at which it can monitor the active CICS region. Depending on how the active CICS region was shut down, the alternate CICS region completes either a warm or emergency restart, if it needs to take over, as follows:

- If the active CICS region was shut down by a successfully completed CEMT PERFORM SHUTDOWN TAKEOVER command, the alternate CICS region performs a **warm** start.
- If the active CICS region was shut down abnormally, the alternate CICS region performs an **emergency** restart.

When it takes over, the alternate CICS region becomes the active CICS region.

The effect of specifying START=STANDBY on the alternate CICS region is similar to that of the START=AUTO option.

If you code START=STANDBY with XRF=NO, initialization fails with message DFHXA6530, and CICS terminates abnormally with a dump.

For information about operating a CICS region with XRF, see the *CICS Operations and Utilities Guide*.

Table 26 shows how the effect of the START parameter depends on the state of the CICS global catalog and system log.

Table 26. Effect of the START= parameter in conjunction with the global catalog and system log

START parm.	State of global catalog	State of system log	Result at restart
Any.	Not defined to VSAM.	Any.	JCL error.
INITIAL	Defined.	Any.	CICS performs an initial start. The global catalog and system log <sup>7</sup> are initialized.
COLD	Defined but contains no recovery manager control record.	Any.	After prompting for confirmation, CICS performs an initial start. The global catalog and system log <sup>7</sup> are initialized.
COLD	Contains recovery manager records.	Not defined or dummy or empty.	Message DFHRM0401 is issued. Startup fails.
COLD	Contains recovery manager records.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Defined but contains no recovery manager control record and no AUTOINIT autostart override.	Any.	Message DFHRM0137 is issued. Startup fails.
AUTO	Contains a recovery manager AUTOINIT autostart override.	Any.	CICS performs an initial start, without prompting. The global catalog and system log <sup>7</sup> are initialized.
AUTO	Contains a recovery manager control record that does <i>not</i> indicate an initial start, and an AUTOCOLD autostart override.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Contains recovery manager records, but no AUTOINIT override.	Not defined or dummy or empty.	Message DFHRM0401 is issued. Startup fails.
AUTO	Contains a recovery manager control record indicating an initial start.	Any.	CICS performs an initial start. The global catalog and system log <sup>7</sup> are initialized.

7. If the system log is defined as a dummy, it is ignored.

Table 26. Effect of the START= parameter in conjunction with the global catalog and system log (continued)

START parm.	State of global catalog	State of system log	Result at restart
AUTO	Contains a recovery manager control record indicating a cold start, and no autostart override.	Contains records from previous run.	CICS performs a cold start. Recovery records in the system log that relate to changes to local resources are deleted.
AUTO	Contains a recovery manager control record indicating an emergency start, and no autostart override.	Contains records from previous run.	CICS performs an emergency start.
AUTO	Contains a recovery manager control record indicating a warm start, and no autostart override.	Contains records from previous run.	CICS performs a warm start.

**Notes:**

1. It is important to keep the CICS global and local catalogs in step. If CICS tries to perform a warm or emergency start and finds that the local catalog has been initialized, startup fails. Therefore, only initialize the local catalog at the same time as the global catalog.
2. It is recommended that you always run the DFHRMUTL and DFHCCUTL utilities in the same job. Run DFHRMUTL first and check its return code before running DFHCCUTL. If you do this, the global and local catalogs should never get out of step. For information about running DFHRMUTL and DFHCCUTL, see the *CICS Operations and Utilities Guide*.

Table 27 shows the effect of different types of CICS startup on the CICS trace, monitoring, statistics, and dump domains.

Table 27. Effect of the type of startup on CICS domains

Domain	State of the CICS catalogs	Warm or emergency start	Initial or cold start
Trace	Not relevant.	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog is newly initialized.	Domain initializes according to the system initialization parameters.	Domain initializes according to the system initialization parameters.
Monitoring	The global catalog contains status of monitoring at the previous CICS shutdown.	Domain uses monitoring status from the catalog, but modified by any system initialization override parameters.	Domain initializes according to the system initialization parameters.
Statistics	The global catalog is newly initialized.	Domain initializes according to CICS-defined system default values.	Domain initializes according to CICS-defined system default values.
Statistics	The global catalog contains status of statistics at CICS shutdown.	Domain uses statistics status from the catalog.	Domain initializes according to CICS-defined system default values.

Table 27. Effect of the type of startup on CICS domains (continued)

Domain	State of the CICS catalogs	Warm or emergency start	Initial or cold start
Dump	The global catalog is newly initialized.	Domain initializes the dump table according to CICS-defined system default values. Other dump attributes are set by system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.
Dump	The global catalog contains dump status at CICS shutdown.	Domain reads the dump table and dump status from the catalog. Other dump attributes are modified by any system initialization parameters.	Domain initializes an empty dump table, and takes CICS-defined default action for all dump requests. Other dump attributes are set by system initialization parameters.

## CICS startup and the VTAM session

In a VTAM network, the session between CICS and VTAM is started automatically if VTAM is started before CICS. If VTAM is not active when you start CICS, you receive the following messages:

```
F vtamname,USERVAR,ID=generic-applid,VALUE=specific-applid
+DFHSI1589D 'applid' VTAM is not currently active.
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=xxxxxxx, ACB CODE=yy.
```

Although the MODIFY NET, USERVAR command is only significant when you are running CICS with XRF, the USERVAR message occurs for both XRF=YES and XRF=NO CICS systems. If you receive messages DFHSI1589D and DFHSI1572, and if the CICS region is not initializing as an alternate CICS region, you can start the CICS-VTAM session manually when VTAM is eventually started, by means of the CEMT SET VTAM OPEN command from a supported MVS console or a non-VTAM terminal.

If VTAM is active, but CICS still cannot open the VTAM ACB because VTAM does not recognize the CICS APPLID, you receive the following messages:

```
F vtamname,USERVAR,ID=generic-applid,VALUE=specific-applid
+DFHSI1592I 'applid' CICS applid not (yet) active to VTAM.
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=00000008, ACB CODE=5A.
```

This may be caused by an error in the value of APPLID operand, in which case you must correct the error and restart CICS. For information about other causes and actions, see the *CICS Messages and Codes*.

### Concurrent initialization of VTAM and XRF alternate CICS regions

An XRF alternate CICS region cannot initialize properly until it has successfully opened the VTAM ACB.

Because VTAM and the alternate CICS region may be initialized concurrently, it is possible that several tries may have to be made to open the VTAM ACB. If VTAM is not active, the following message is written to the system console every 15 seconds:

```
DFHSI1589D 'applid' VTAM is not currently active.
```

If VTAM is active, but CICS cannot open the VTAM ACB, the following messages are written to the system console:

```
+DFHSI1572 'applid' Unable to OPEN VTAM ACB - RC=xxxxxxx, ACB CODE=yy.  
DFHSI1590 'applid' XRF alternate cannot proceed without VTAM.
```

CICS abends with a dump (abend code 1590).

## The CICS parameter manager domain

In addition to loading the system initialization table at the start of initialization, and reading any other parameters from PARM, SYSIN, or the system console, the parameter manager domain is responsible for the management of the SIT. With the exception of the application domain (AP) which uses the SIT directly, the parameter manager domain passes system initialization parameters to the other CICS domains on request.

The domain initialization process is as follows:

### Query the type of startup

With the exception of the trace domain, each domain asks the parameter manager for the type of startup—initial, cold, or warm. (For this purpose, the parameter manager domain treats an emergency restart as a warm start.)

### Startup is initial or cold

If startup is initial or cold, domains do not read their domain records from the catalogs. Instead, they request system initialization parameters from the parameter manager domain. Because it is a cold start, the parameter manager domain returns all system initialization parameters from the SIT, modified by any overrides.

### Startup is warm

If startup is warm, domains try to perform a warm start by reading their domain records from the catalogs:

- If they succeed in reading their status records, domains perform a warm start. Where applicable, they also request system initialization parameters from the parameter manager domain. Because it is a warm start, the parameter manager domain returns only those system initialization parameters supplied as overrides by PARM, SYSIN, or the system console.
- If they fail for any reason to read their status records, domains perform a cold start. They do this either by requesting **all** system initialization parameters from the parameter manager domain, or by using system default values if the domain does not have any system initialization parameters.

### NEWSIT or new suffix

Although a START=AUTO may resolve to a warm start, parameter manager enforces most system initialization parameters if:

- You specify NEWSIT=YES as a system initialization parameter in PARM, SYSIN, or through the console.
- You specify a different SIT suffix from the previous run of CICS. Parameter manager saves the suffix from each run in the global catalog, and, if it detects a new suffix, it forces the NEWSIT=YES option.

For details of the parameters that are ignored when you specify NEWSIT=YES, see the NEWSIT parameter description on page “NEWSIT” on page 218.

**Note:** The trace domain is an exception to the above rules in that it always cold starts. Trace does not save its status at CICS shutdown like the other

domains, and regardless of the type of startup, it requests **all** of its system initialization parameters from the parameter manager domain.

## End of CICS startup

Whichever type of startup is performed, when the message:

```
DFHSI1517 - 'applid': Control is being given to CICS.
```

is displayed on the operating system console, CICS is ready to process terminal requests. (*applid* is the value of the specific APPLID system initialization parameter.)

When the startup process is completed, users are able to enter transactions from any terminals that are connected to CICS. For information about the CICS-supplied transactions, see *CICS Supplied Transactions*.

---

## Chapter 20. Defining the JVM options (JVM profiles and JVM properties files)

This chapter describes the options that you can specify for the initialization of an IBM persistent reusable Java<sup>®</sup> Virtual Machine (JVM) under CICS.

You specify options for the JVM in a JVM profile, which is named by the program resource definition of a Java application program that requires a JVM with these options. JVM profiles are text files stored on HFS. Each JVM profile references a JVM properties file, which is another text file containing the system properties for the JVM. You can edit JVM profiles and JVM properties files using any standard text editor.

"How CICS creates JVMs" in *Java Applications in CICS* explains how CICS uses JVM profiles, and gives an overview of the options that you can specify using JVM profiles and JVM properties files. "Setting up JVM profiles and JVM properties files" in *Java Applications in CICS* tells you how to set up suitable JVM profiles and JVM properties files to meet the needs of your applications. To help you with that task, this chapter provides the following reference information:

- "Rules for coding JVM profiles and JVM properties files," which you should read before coding these files.
- "Options in JVM profiles" on page 290, which lists the options you can specify in a JVM profile.
- "System properties for JVMs" on page 310, which lists the options you can specify in a JVM properties file.
- "The sample JVM profiles and JVM properties files" on page 321, which provides the full text of the CICS-supplied sample files.

---

### Rules for coding JVM profiles and JVM properties files

The name of a JVM profile can be up to 8 characters in length. The name of a JVM properties file can be any length, but for ease of use, choose either the name of the JVM profile that references it, or another short name.

The name of a JVM profile or JVM properties file can include the following characters:

**Acceptable characters**

A-Z a-z 0-9 @ # . - \_ % & \$ ? ! : | " = , ; < >

When creating your own JVM profile or JVM properties file, do not give it a name beginning with DFH, because these characters are reserved for use by CICS.

As JVM profiles and JVM properties files are HFS files, case is important. When you specify the name of a JVM profile or JVM properties file anywhere in CICS, you must enter it using the same combination of upper and lower case characters that is present in the HFS file name. The CEDA panels accept mixed case input for the JVMPROFILE field irrespective of your terminal's UCTRAN setting. However, this does not apply when values for this field are supplied on the CEDA command line, or when you are using another CICS transaction such as CEMT or CECI. If you need to enter the name of a JVM profile in mixed case when you use CEDA from

the command line or when you use another CICS transaction, ensure that the terminal you use is correctly configured, with upper case translation suppressed.

Options in a JVM profile, or system properties in a JVM properties file, take the form of a keyword and value separated by an = sign. For example:

```
VERBOSE=NO  
ibm.jvm.events.output=event.log
```

Each JVM option or system property is therefore a name and value element pair.

Only the JVM options listed in “Options in JVM profiles” on page 290 are recognized by CICS for use in a JVM profile. However, there is no such restriction for a JVM properties file. CICS passes all the system properties in a JVM properties file to the JVM unchanged. You should bear in mind that only the system properties described in the CICS documentation are supported by CICS, although the JVM can support a much wider range of system properties. “System properties for JVMs” on page 310 documents the system properties that are particularly relevant for JVMs in a CICS environment, including some which are defined by CICS.

*Persistent Reusable Java Virtual Machine User's Guide*, SC34-6201, lists command-line options, JVM options and system properties that are used in a persistent reusable JVM in a z/OS environment, some of which are provided in a different format by options in the JVM profile in CICS. The *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/](http://www.ibm.com/developerworks/java/jdk/diagnosis/) documents other system properties that are used for JVM trace and problem determination. The Java class libraries include other system properties, and applications might have their own system properties. There is no central repository of all system properties for the JVM.

Follow these rules when coding JVM profiles and JVM properties files:

#### **Case sensitivity**

All parameter keywords and operands specified in a JVM profile or JVM properties file are case-sensitive, and must be specified exactly as shown in “Options in JVM profiles” on page 290 and “System properties for JVMs” on page 310.

#### **Class path separator character**

Use the : (colon) character to separate the directory paths that you specify on a class path option. The separator character is defined by the `path.separator` system property in the JVM properties file, and the colon is the default for this system property, but you can change this. The options that you might use to specify class paths are `LIBPATH`, `CLASSPATH`, `TMPREFIX` and `TMSUFFIX` in the JVM profile, and `ibm.jvm.shareable.application.class.path` in the JVM properties file.

#### **Continuation**

For JVM options or system properties, the value is delimited by the end of the line in the text file. If a system property or JVM option, such as a class path, that you are entering or editing is too long for an editor window, you can break the line to avoid scrolling. To continue on the next line, terminate the current line with the backslash and a blank (\ ) continuation characters. For example, the `LIBPATH` option in the CICS-supplied sample JVM profiles is coded as follows:

```
LIBPATH=\
  /usr/lpp/cicsts/cicsts31/lib:\
  /usr/lpp/cicsts/cicsts31/ctg:\
  /usr/lpp/java142/J1.4/bin:\
  /usr/lpp/java142/J1.4/bin/classic
```

### Comments

To add comments in a JVM profile or JVM properties file, precede the comment with a # symbol. Comment lines are ignored when the file is read by the JVM launcher. For example, the following lines are comments taken from the sample JVM properties file `dfjjvmp.r.props`:

```
# Uncomment the following line to specify a classpath
# for Java classes that are CICS programs or Corba
# applications, but not EJB jars.
#
# ibm.jvm.shareable.application.class.path=user.jar:user.directory
```

JVM profiles and JVM properties files can also contain blank lines, which are also ignored. You can use blank lines as a separator between options or groups of options.

### Character escape sequences

Within a property element string, you can code the escape sequences shown in Table 28

Table 28. Escape sequences

Escape sequence	Character value
<code>\b</code>	Backspace
<code>\t</code>	Horizontal tab
<code>\n</code>	Newline
<code>\r</code>	Carriage return
<code>\"</code>	Double quote
<code>\'</code>	Single quote
<code>\\</code>	Backslash
<code>\xxx</code>	The character corresponding to the octal value <code>xxx</code> , where <code>xxx</code> is between 000 and 377
<code>\uxxxx</code>	The Unicode character with encoding <code>xxxx</code> , where <code>xxxx</code> is one to four hexadecimal digits (see note below for more information).

**Note:** Unicode `\u` escapes are distinct from the other escape types. The Unicode escape sequences are processed before the other escape sequences described in Table 28. A Unicode escape is simply an alternative way to represent a character that may not be displayable on non-Unicode systems. The character escapes, however, can represent special characters in a way that prevents the usual interpretation of those characters.

### Storage sizes

When specifying Java nonstandard options in a JVM profile, specify storage sizes in multiples of 1024 bytes. Use the letter K to indicate kilobytes, the letter M to indicate megabytes, and the letter G to indicate gigabytes. For example, to specify 6291456 bytes as the initial size of the middleware heap, code `Xms` in one of the following ways:

Xms=6291456  
Xms=6144K  
Xms=6M

---

## Options in JVM profiles

"Setting up JVM profiles and JVM properties files" in *Java Applications in CICS* tells you how to set up suitable JVM profiles and JVM properties files to meet the needs of your applications.

The options that you can specify in JVM profiles can be divided into three groups:

- Those that are required only by CICS.
- The standard JVM options used when launching a JVM.
- A set of nonstandard options, some of which are used for tuning, and others for debugging in a development environment.

Table 29 on page 293 summarizes the options according to their function. Specify each option according to the coding rules described in "Rules for coding JVM profiles and JVM properties files" on page 287.

### Notes:

1. Remember that all parameter keywords and operands specified in a JVM profile or JVM properties file are case-sensitive, and must be specified exactly as shown in the following sections.
2. Where a default value is indicated for a particular option, this is the default value that CICS uses when the option is **not** specified in the JVM profile. However, some or all of the CICS-supplied sample JVM profiles might specify a value that is different to the default value. For example, the default value for the `Xverify` option is `remote`, but the CICS-supplied sample JVM profiles specify the value `none`.

### JVM profiles that specify CLASSCACHE=YES (for worker JVMs):

If you specify `CLASSCACHE=YES` in a JVM profile, the JVM uses the shared class cache, and is known as a worker JVM. *Java Applications in CICS* describes the shared class cache.

When you specify `CLASSCACHE=YES` in a JVM profile, certain options in the JVM profile and JVM properties file are ignored. If these options are found in the JVM profile or JVM properties file for a worker JVM, CICS does not pass them on to the JVM. If values for these options are required, they are taken from the JVM profile and JVM properties file for the master JVM that initializes the shared class cache. These options are not used in the CICS-supplied sample worker JVM profile `DFHJVMP`. If you have converted another JVM profile to use the shared class cache, you can either remove the options (by commenting out or deletion) from the JVM profile or JVM properties file, or leave them there.

The options that are ignored because you specify `CLASSCACHE=YES` in a JVM profile are:

- `LIBPATH` in the JVM profile, which specifies the library path, `CICS_DIRECTORY` and `JAVA_HOME` in the JVM profile, which specify the CICS and Java directories on HFS, and `TMPREFIX` and `TMSUFFIX` in the JVM profile, which can be used to specify the trusted middleware class path. If you need to specify additional directory paths for a worker JVM, use the corresponding options in the JVM profile for the master JVM.

- Xdebug in the JVM profile, which enables debugging support in the JVM. (Worker JVMs cannot be configured for debug.)
- Xinitacsh and Xinitsh in the JVM profile, which specify sizes for the system heap. (Worker JVMs share the master JVM's system heap.)
- REUSE and the older option Xresettable in the JVM profile, either of which defines the level of reusability for the JVM. (Worker JVMs inherit their level of reusability from the master JVM.)
- The `ibm.jvm.shareable.application.class.path` system property in the JVM properties file, which specifies the shareable application class path. To specify the directory paths for application classes that run in a worker JVM, use the `ibm.jvm.shareable.application.class.path` system property in the JVM properties file for the master JVM.

The `java.compiler` system property in the JVM properties file, which names the Java just-in-time compiler, is a special case. You can use this system property to specify that Java compiling should not be performed for a particular worker JVM, but otherwise it must match the `java.compiler` system property for the master JVM.

The following sections of this topic have more information about the JVM profile options in the list above, and the topic “System properties for JVMs” on page 310 has more information about the system properties.

#### **JVM profile for the master JVM that initializes the shared class cache:**

The JVM profile that is used for the master JVM that initializes the shared class cache is specified by the `JVMCCPROFILE` system initialization parameter. It can be changed using the `PROFILE` option on the `CEMT PERFORM CLASSSCACHE START` or `CEMT PERFORM CLASSSCACHE RELOAD` commands (or the equivalent `EXEC CICS` commands). *Java Applications in CICS* has more information about determining the JVM profile that is used for the master JVM. The CICS-supplied sample JVM profile for the master JVM is `DFHJVMCC`, and the JVM properties file that it references is `dfjjvmcc.props`.

The JVM profile for a master JVM is similar to the JVM profile for any other JVM. The `CLASSSCACHE_MSGLOG` option can be specified to name the file for messages from the master JVM. One important decision to make about the master JVM is whether to define it as a resettable JVM (by specifying the option `REUSE=RESET`), or as a continuous JVM (by specifying the option `REUSE=YES`). It cannot be defined as a single-use JVM (`REUSE=NO`). The worker JVMs in a CICS region all have the same level of reusability as the master JVM in that region. *Java Applications in CICS* has more information to help you decide on an appropriate level of reusability for your worker JVMs.

Also, CICS ignores certain options in the JVM profile which are not needed for a master JVM, as follows:

- `CLASSSCACHE`, which makes a JVM into a worker JVM.
- `CLASSPATH`, which specifies the standard class path containing nonshareable application classes. (Applications cannot run in a master JVM.)
- `USEROUTPUTCLASS`, which names a class to be used to redirect output from the JVM. Although CICS does recognize this option for a master JVM, the output redirection class will never be invoked by the activities of the master JVM, so it is irrelevant.
- Xdebug, which enables debugging support in the JVM. (A master JVM cannot be configured for debug.)

- `Xinitth`, which specifies the initial transient heap size. (A master JVM does not have a transient heap.)

The JVM properties file for a master JVM omits most of the system properties that would be specified for a normal JVM, because the master JVM is not used to run Java applications. The only system property that needs to be specified is `ibm.jvm.shareable.application.class.path`, which you should use to specify the shareable application classes for all the applications that will run in worker JVMs that use the shared class cache. You might also want to specify the system properties `java.compiler` and `ibm.dg.trc.external`, and the topic “System properties for JVMs” on page 310 has information about these.

### Customizing DFHJVMCD:

The JVM profile DFHJVMCD is reserved for use by CICS-supplied system programs, in particular the default request processor program DFJIIRP (used by the CICS-supplied CIRP request processor transaction), to make them independent of any changes you make to the default JVM profile DFHJMMPR. DFHJVMCD has an associated JVM properties file, `dfjjvmcd.props`. Do not specify this profile in PROGRAM resource definitions that you set up for your own applications.

You need to make sure that DFHJVMCD is set up correctly for your CICS region, but you should customize it only where necessary. The options that you can change are indicated in the text of DFHJVMCD. You might need to change options in DFHJVMCD if:

- You want JVMs created using DFHJVMCD to use the shared class cache. The default is that they are standalone JVMs.
- You want to change the level of reusability for JVMs created using DFHJVMCD. The default is that they are continuous JVMs. You can change them to resettable JVMs, if they need to use the shared class cache, and your other worker JVMs are resettable JVMs. Do not change them to single-use JVMs.
- You need to change the library path, the `CICS_DIRECTORY` or `JAVA_HOME` options, or the directory containing the JVM properties files, to match your installation's requirements.
- You want to change the default locations for output from JVMs created using DFHJVMCD, or to redirect the output using the `USEROUTPUTCLASS` option.
- You want to tune the heap sizes for JVMs created using DFHJVMCD.
- You have enterprise beans that use JDBC. In this case, the relevant DB2 libraries and files (as specified in the sample profile DFHJMMPR) must be added to the `LIBPATH` and `TMPREFIX` settings for DFHJVMCD.

In `dfjjvmcd.props`, you need to add the paths to classes for the applications using the request processor program to the shareable application class path (except for the deployed JAR files for enterprise beans, which do not need to be added to a class path). You also need to specify the system properties that are necessary to configure your JNDI nameserver and LDAP nameserver, and enable the Java 2 security policy mechanism (if required). “System properties for JVMs” on page 310 has more information about the relevant system properties to change in `dfjjvmcd.props`.

Do not make any other changes to DFHJVMCD and `dfjjvmcd.props`.

Table 29 on page 293 summarizes the main options that you can use in a JVM profile for CICS, and also some of the system properties that are normally suitable

for a JVM properties file in a CICS environment. There are other possible system properties for JVMs which are not included in this table. Some are not included in the table because it is recommended that you should not normally change them, and these are listed in “System properties for JVMs” on page 310. Others are not included in the table, and are not listed in this document, because they are not normally relevant for a CICS environment. Remember that as noted in “Rules for coding JVM profiles and JVM properties files” on page 287, other documentation relating to the JVM lists further system properties, but only the system properties described in the CICS documentation are supported by CICS.

The table shows the default for each option if it is not specified in the JVM profile or JVM properties file. It also normally shows, for each type of JVM, whether the option is required (must be specified), OK (is valid), or ignored (CICS ignores the option if it is specified). If a particular setting for the option is required for a certain type of JVM, or if the option is only applicable to a certain type of JVM, then this information is given instead. The full description of each JVM profile option is in the list following the table, and the full description of the system properties mentioned here is in “System properties for JVMs” on page 310.

Table 29. Main options for a JVM profile and JVM properties file in a CICS environment

Option	Default	Where set	Master	Worker	Standalone
<i>JVM type</i>					
CLASSCACHE	NO	Profile	Ignored	YES	NO
REUSE	RESET	Profile	RESET or YES	Ignored	Any
Xresettable (deprecated)	YES	Profile	YES	Ignored	Any
<i>Directories</i>					
CICS_DIRECTORY	none	Profile	Required	Ignored	Required
JAVA_HOME	none	Profile	Required	Ignored	Required
WORK_DIR	/tmp	Profile	OK	OK	OK
<i>Class paths</i>					
CLASSPATH	none	Profile	Ignored	OK	OK
LIBPATH	none	Profile	Required	Ignored	Required
TMPREFIX, TMSUFFIX	none	Profile	OK	Ignored	OK
ibm.jvm.shareable.application.class.path	none	Properties	Required	Ignored	OK
<i>Further settings and facilities for the JVM</i>					
JVMPROPS	none	Profile	Required	Required	Required
INVOKE_DFHJVMAT	NO	Profile	Ignored	Ignored	Single-use only
java.compiler	jitc	Properties	OK	OK	OK
MAX_RESETS_TO_GC	100	Profile	Ignored	OK	Resettable only
Xrundllname	none	Profile	OK	OK	OK
<i>Storage heap sizes</i>					
Xinitacsh	128 KB	Profile	OK	Ignored	Resettable only
Xinitsh	128 KB	Profile	OK	Ignored	OK

#  
#

Table 29. Main options for a JVM profile and JVM properties file in a CICS environment (continued)

Option	Default	Where set	Master	Worker	Standalone
Xinitth	500 KB	Profile	Ignored	Resettable only	Resettable only
Xmaxe, Xmaxf, Xmine, Xminf	see listing	Profile	OK	OK	OK
Xms	500 KB	Profile	OK	OK	OK
Xmx	64 MB	Profile	OK	OK	OK
Xoss, Xss	see listing	Profile	OK	OK	OK
<i>Output from the JVM</i>					
CLASSCACHE_MSGLOG	dfhjvmccmsg.log	Profile	Required	Ignored	Ignored
LEHEAPSTATS	NO	Profile	OK	OK	OK
STDERR	dfhjvmerr	Profile	OK	OK	OK
STDIN	dfhjvmin	Profile	OK	OK	OK
STDOUT	dfhjvmout	Profile	OK	OK	OK
USEROUTPUTCLASS	none	Profile	Ignored	OK	OK
VERBOSE	NO	Profile	OK	OK	OK
<i>Security</i>					
java.security.manager	none	Properties	OK	OK	OK
java.security.policy	none	Properties	OK	OK	OK
<i>Name server configuration</i>					
com.ibm.cics.ejs.nameserver	none	Properties	OK	OK	OK
com.ibm.ws.naming.ldap.containerdn, com.ibm.ws.naming.ldap.noderootrdn	none	Properties	OK	OK	OK
java.naming.security.authentication, java.naming.security.credentials, java.naming.security.principal	none	Properties	OK	OK	OK
<i>JDBC</i>					
com.ibm.cics.datasource.path	none	Properties	OK	OK	OK
jdbc.drivers	none	Properties	OK	OK	OK
<i>Problem determination and application debugging</i>					
DISABLEASSERTIONS	none	Profile	OK	OK	OK
ENABLEASSERTIONS	none	Profile	OK	OK	OK
SYSTEMASSERTIONS	none	Profile	OK	Error	OK
USE_LIBJVM_G	NO	Profile	OK	OK	OK
Xcheck	NO	Profile	OK	OK	OK
Xdebug	NO	Profile	Ignored	Ignored	OK
Xverify	remote	Profile	OK	OK	OK
ibm.dg.trc.external	none	Properties	OK	OK	OK
ibm.jvm.crossheap.events	none	Properties	OK	OK	OK
ibm.jvm.events.output	event.log	Properties	OK	OK	OK
ibm.jvm.resettrace.events	none	Properties	OK	OK	OK
ibm.jvm.unresettable.events.level	none	Properties	OK	OK	OK

Table 29. Main options for a JVM profile and JVM properties file in a CICS environment (continued)

Option	Default	Where set	Master	Worker	Standalone
gateway.T properties (for tracing CICS Connector for CICS TS)	none	Properties	OK	OK	OK

## CICS-specific options

This set of JVM profile options is required only by CICS to enable CICS to start the JVM.

**CICS\_DIRECTORY=/usr/lpp/cicsts/cicsts31/**

Specifies the path for the CICS HFS /lib subdirectory containing the CICS-supplied JAR files, such as dfjcics.jar, dfjorb.jar, ras.jar and so on. By default, /lib is installed in /usr/lpp/cicsts/cicsts31/, where *cicsts31* is defined by the USSDIR installation parameter when you installed CICS TS for z/OS, Version 3.1. For more information about this installation parameter, see the description of the &CICS\_DIRECTORY symbol in the *CICS Transaction Server for z/OS Installation Guide*.

This parameter is required in JVM profiles where you have specified **CLASSCACHE=NO** (or where you have not specified the CLASSCACHE option at all). It is used by CICS to build the trusted middleware class path.

In JVM profiles where you have specified **CLASSCACHE=YES** (that is, the JVM profiles for worker JVMs), this option is ignored. Its value is taken instead from the JVM profile for the master JVM that initializes the shared class cache. The CICS-supplied sample JVM profile for the master JVM is DFHJVMCC.

**CLASSCACHE={YES|NO}**

Specifies whether this JVM is to use the shared class cache. The default is NO. *Java Applications in CICS* describes the shared class cache. A JVM that uses the shared class cache is known as a worker JVM. Single-use JVMs or JVMs that are configured for debug cannot use the shared class cache.

When you specify CLASSCACHE=YES in a JVM profile, certain options in the JVM profile and JVM properties file are ignored. If these options are found in the JVM profile or JVM properties file for a worker JVM, CICS does not pass them on to the JVM. If values for these options are required, they are taken from the JVM profile and JVM properties file for the master JVM that initializes the shared class cache. The introduction to this topic explains which options and system properties are affected.

The CLASSCACHE option is ignored if it is specified in the JVM profile used for the master JVM that initializes the shared class cache.

In JVM profiles where you have specified CLASSCACHE=YES, in a production environment, you should avoid using the CLASSPATH option (the standard class path) to specify application classes. If you do use this option, if the JVM is resettable, the classes are reloaded each time the JVM is reused. This is detrimental to performance, and means that the JIT compiler will generate less-than-optimal code for those classes. Instead, you should place the application classes on the `ibm.jvm.shareable.application.class.path` system property in the JVM properties file that is used for the master JVM that initializes the shared class cache.

**CLASSCACHE\_MSGLOG={dfhjvmccmsg.1og|filename}**

Specifies the name of an HFS file allocated in WORK\_DIR, to which messages are written from the master JVM that initializes the shared class cache. If no file name is specified, or if the option is not included, the default file name

dfhjvmccmsg.log is used. This option is only used on the JVM profile for the master JVM that initializes the shared class cache. If it is specified on any other JVM profile, it is ignored.

**INVOKE\_DFHJVMAT={NO|YES}**

Specifies whether or not the user replaceable module, DFHJVMAT, should be invoked before creating a new JVM. DFHJVMAT can only be used for single-use JVMs, that is, where the option REUSE=NO or the older option Xresettable=NO is specified in the JVM profile. INVOKE\_DFHJVMAT is ignored if the JVM is a resettable JVM or a continuous JVM, which is the case if any of the following options are specified in the JVM profile:

- REUSE=RESET or REUSE=YES
- Xresettable=YES
- CLASSCACHE=YES

**JAVA\_HOME=/usr/lpp/java142/J1.4/**

Specifies the path for IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2 subdirectories and JAR files. By default, the Java subdirectories and JAR files are installed in /usr/lpp/java142/J1.4/, where /java142/J1.4/ is defined when you install IBM Software Developer Kit for z/OS, Java 2 Technology Edition, Version 1.4.2.

The JAVA\_HOME directory is also the value you specify on the JAVADIR CICS installation parameter. For more information, see the description of the &JAVA\_HOME symbol in the *CICS Transaction Server for z/OS Installation Guide*.

This parameter is required in JVM profiles where you have specified **CLASSCACHE=NO** (or where you have not specified the CLASSCACHE option at all). In JVM profiles where you have specified **CLASSCACHE=YES** (that is, the JVM profiles for worker JVMs), this option is ignored. Its value is taken instead from the JVM profile for the master JVM that initializes the shared class cache. The CICS-supplied sample JVM profile for the master JVM is DFHJVMCC.

**JVMPROPS=path/file\_name**

Specifies the path and name of a JVM properties file, which is a HFS file that contains the system properties for this JVM. The values in this file are passed to the JVM for the construction of system properties as if specified by a -D option in a java command. “System properties for JVMs” on page 310 tells you what you can specify in a JVM properties file.

**LEHEAPSTATS={YES|NO}**

Specifies whether or not statistics are to be collected for the amount of Language Environment heap storage that is used by the JVM. The default is NO. The statistics appear as the field “Peak Language Environment heap storage used” in the JVM Profile statistics. Collecting these statistics affects the performance of the JVM, so you should only specify **LEHEAPSTATS=YES** while you are in the process of tuning the heap sizes for your JVMs. (“Java applications using a Java virtual machine (JVM): improving performance” in the *CICS Performance Guide* explains this process.) In a production environment, you should specify **LEHEAPSTATS=NO**.

**LIBPATH=pathname**

Specifies the directory paths to be searched for native C dynamic link library (DLL) files that are used by the JVM (which have the extension .so in z/OS UNIX), including those required to run the JVM and additional native libraries loaded by trusted code.

The directory paths set in the CICS-supplied sample JVM profiles are created automatically using the symbols &CICS\_DIRECTORY and &JAVA\_HOME. The DFHIJVMJ job substitutes appropriate values for these symbols during CICS installation (see the *CICS Transaction Server for z/OS Installation Guide*). The directory paths that are created by this process include the path to the native C DLL files required to support JCICS.

If CLASSCACHE=YES is specified to make the JVM a worker JVM, this option is ignored. Its value is taken instead from the library path specified in the JVM profile for the master JVM that initializes the shared class cache. If you have added directory paths (for example, the DB2-supplied directory containing dynamic load libraries for the JDBC drivers) to this option in the JVM profile for a worker JVM, ensure that you transfer them to the corresponding option in the JVM profile for the master JVM. The CICS-supplied sample JVM profile for the master JVM is DFHJVMCC. Note that the master and worker JVMs use the same LIBPATH specification to ensure that they are using the same versions of the files on the library path. However, these files are not loaded into the shared class cache. Unless they are shared through another z/OS facility (such as the shared library region), a copy is loaded into each worker JVM.

#  
#  
#  
#

**MAX\_RESETS\_TO\_GC={100|number}**

Specifies when the garbage collection cycle is started in CICS. The value is the number of times a JVM is used. The default is 100, and you can set a minimum value of 1 and a maximum value of 2147483647 ( $2^{31}-1$ ).

**REUSE={RESET|YES|NO}**

Specifies the level of reusability for the JVM. “How JVMs are reused and reset” in *Java Applications in CICS* gives a full explanation of the characteristics of the three possible levels of reusability for a JVM, the situations and application designs for which each level of reusability is appropriate, and the relative performance of each level of reusability. The settings for this option specify the levels of reusability as follows:

- **REUSE=RESET**, which is the default, creates a JVM that is **reused** many times by Java applications, and is **reset** after each Java program has completed. This type of JVM is known as a resettable JVM.
- **REUSE=YES** creates a JVM that is **reused** many times by Java applications, but is **not reset** after each Java program has completed. This type of JVM is known as a continuous JVM.
- **REUSE=NO** creates a JVM that is like the earlier JVM that was supported by CICS in CICS TS 1.3. It is **not reused** and **not reset**, but instead is destroyed after a single Java program has run in it. This type of JVM is known as a single-use JVM.

The level of reusability that you choose might affect the relevance of some of the storage heap size options in the JVM profile. The options that might be affected are Xinitacsh, Xinitth, Xmaxe, Xmine, Xmaxf, Xminf, and Xmx.

Depending on the level of reusability, you might be able to omit these options from the JVM profile, or their meaning might change. See the descriptions of these options later in this section for more information.

When you are choosing a setting for the REUSE option in a JVM profile, follow these rules:

1. If the option **CLASSCACHE=YES** is specified in the JVM profile, to make the JVM a worker JVM, the REUSE option is ignored. Worker JVMs inherit their level of reusability from the master JVM. Do not specify both **CLASSCACHE=YES** and the REUSE option.

2. If the JVM profile is to be used for the master JVM that initializes the shared class cache, it can specify **REUSE=RESET** (to create a resettable JVM) or **REUSE=YES** (to create a continuous JVM). The master JVM cannot be a single-use JVM (**REUSE=NO**), and if you specify that setting, CICS disallows it and assumes instead that the master JVM is resettable. If the REUSE option is omitted, CICS also assumes that the master JVM is resettable. The worker JVMs in a CICS region all inherit their level of reusability from the master JVM in that region.
3. If you specify the REUSE option, it is advisable to remove the older option `Xresettable` from the JVM profile, if it is present. This option was the previous method of specifying the level of reusability for a JVM. For migration purposes, it is still a valid way to create a resettable JVM (which is specified by `Xresettable=YES`) or a single-use JVM (which is specified by `Xresettable=NO`), but it cannot be used to create a continuous JVM. If both the `Xresettable` option and the REUSE option are specified, an error message is issued if their settings conflict, but the REUSE option overrides the `Xresettable` option. The `Xresettable` option is not used in the CICS-supplied sample JVM profiles.

If you specify **REUSE=RESET** or **REUSE=YES** in a JVM profile, remember that for successful reuse of JVMs created using that profile, you should:

- Ensure that only trusted middleware classes (normally middleware supplied by IBM or another vendor) are placed on the trusted middleware class path, and your own application classes are placed on the shareable application class path or the standard class path. “Classes in a JVM” in *Java Applications in CICS* explains the difference between middleware and application classes and class paths.
- If you choose **REUSE=RESET**, ensure that applications do not cause the JVM to be made unresettable. In a resettable JVM, application classes must follow a strict set of rules which, if observed, ensure that the JVM can be properly reset. The restricted actions are not prevented by the JVM, but they are detected and cause the JVM to be destroyed at termination of the Java application. “How JVMs are reused and reset” in *Java Applications in CICS* explains more about this process, and *Persistent Reusable Java Virtual Machine User's Guide*, SC34-6201, has details of the unresettable actions.
- If you choose **REUSE=YES**, ensure that when designing applications to run in a continuous JVM, you have taken into account the programming considerations described in “How JVMs are reused and reset” in *Java Applications in CICS*. Because this type of JVM does not apply the same restrictions to application classes as the resettable JVM, you need to make sure that the applications do not change the state of the JVM in undesirable ways, or accidentally leave any unwanted state in the JVM.

**STDERR={dfhjvmerr|file\_name} [ -generate]**

Specifies the name of the HFS file to be used for stderr. If the file does not exist, it is created in the directory specified by the `WORK_DIR` option. If the file already exists, output is appended to the end of the file. On termination of the JVM, if the stderr file is empty and it has been generated for the specific JVM, it is deleted. Otherwise, the file is kept.

The default name for the file is `dfhjvmerr`. Note that for a fixed file name, the output from multiple JVMs is appended to the named file, and the output is interleaved.

You can use in your file name the `&APPLID` symbol, for which CICS substitutes the actual CICS region `APPLID`.

### **-generate**

Specifies that you want CICS to generate the output file name for a specific JVM by appending the following qualifiers to the name supplied on the generate option:

*region* The applid of the CICS region

*time* The current time in the form yydddhhmmss.

**.txt** A literal string suffix to indicate that the file contains readable data and should be transferred with character translation by tools such as FTP.

**-generate** must be preceded by one blank.

The use of the **-generate** option is not recommended in a production environment, because it can be detrimental to the performance of your JVMs.

If you specify the USEROUTPUTCLASS option on a JVM profile, the Java class named on that option handles the System.err requests instead. The HFS file named by the STDERR option could still be used if the class named by the USEROUTPUTCLASS option is unable to write data to its intended destination. This is the case when you use the CICS-supplied sample class com.ibm.cics.samples.SJMergedStream. The file could also be used if output is directed to it for any other reason by a class named by the USEROUTPUTCLASS option.

**STDIN={dfhjvmin|file\_name}**

Specifies the name of the HFS file to be used for stdin. If the file does not exist, it is created in the directory specified by the WORK\_DIR option.

**STDOUT={dfhjvmout|file\_name} [ -generate]**

Specifies the name of the HFS file that is to be used for output to the stdout file. If the file does not exist, it is created in the directory specified by the WORK\_DIR option. If the file already exists, output is appended to the end of the file. On termination of the JVM, if the stdout file is empty and it has been generated for the specific JVM, it is deleted. Otherwise, the file is kept.

The default name for the file is dfhjvmout. Note that for a fixed file name, the output from multiple JVMs is appended to the named file, and the output is interleaved.

You can use in your file name the &APPLID; symbol, for which CICS substitutes the actual CICS region APPLID.

For details of the **-generate** option, see the STDERR option. **-generate** must be preceded by one blank.

If you specify the USEROUTPUTCLASS option on a JVM profile, the Java class named on that option handles the System.out requests instead. The HFS file named by the STDOUT option could still be used if the class named by the USEROUTPUTCLASS option is unable to write data to its intended destination. This is the case when you use the CICS-supplied sample class com.ibm.cics.samples.SJMergedStream. The file could also be used if output is directed to it for any other reason by a class named by the USEROUTPUTCLASS option.

**TMPREFIX=path\_name, TMSUFFIX=path\_name**

Specify paths to be added to the trusted middleware class path that CICS generates automatically from the CICS\_DIRECTORY parameter. The paths specified on TMPREFIX are inserted at the *beginning* of the CICS-generated ibm.jvm.trusted.middleware.class.path system property. The paths specified

on TMSUFFIX are added to the *end* of the CICS-generated `ibm.jvm.trusted.middleware.class.path` system property. Do not edit the `ibm.jvm.trusted.middleware.class.path` system property directly.

If you want to access DB2 from a Java application program using the SQLJ and JDBC APIs, you need to add a DB2-supplied zip file (or for the DB2 Universal JDBC Driver, three DB2-supplied jar files) to the trusted middleware class path. “Using JDBC and SQLJ to access DB2 data from Java programs and enterprise beans written for CICS” in the *CICS DB2 Guide* explains how to do this for various levels of the DB2-supplied JDBC drivers.

Middleware classes have privileges that are not available to the application, and which enable optimizations through the caching of state (loading of classes and native libraries, for example) to be used by multiple applications, even if the JVM is resettable. However, middleware is also responsible for resetting itself correctly at the end of a transaction and, if necessary, for reinitializing at the beginning of a new transaction, in order to isolate different applications from each other.

You might need to specify paths to classes for middleware supplied by IBM or by another vendor, which are not included in the standard JVM setup for CICS. However, you should not use the trusted middleware class path parameters to specify paths to your own application classes, because you cannot subject trusted middleware classes to the same restrictions that might be appropriate for application classes. Classes on the trusted middleware class path could make changes to the JVM environment, or could create objects in one invocation of an application that could persist and interfere with the next invocation, or with the invocation of a completely different application. This is the case even when the JVM is resettable, and you might not want to permit your own application classes to perform these actions. Place your own application classes on the shareable application class path or the standard class path, because then you can use the REUSE option in the JVM profile to restrict the scope of their activities as necessary.

In JVM profiles where you have specified **CLASSCACHE=YES** (that is, the JVM profiles for worker JVMs), the options relating to the trusted middleware class path are ignored. Their value is taken instead from the JVM profile for the master JVM that initializes the shared class cache. Any paths that you have specified using these options in the JVM profiles for worker JVMs, must be copied to the TMPREFIX or TMSUFFIX option in the JVM profile for the master JVM. The CICS-supplied sample JVM profile for the master JVM is DFHJVMCC.

#### **USE\_LIBJVM\_G={YES|NO}**

Specifying **USE\_LIBJVM\_G=YES** enables the debug libraries for the JVM. If you specify NO or omit the option, the optimized libraries are used. Note that when you are using the debug libraries, extra checking takes place, so performance is greatly reduced compared to the use of the the normal, optimized libraries.

This option should be used only under the direction of IBM service.

#### **USEROUTPUTCLASS={classname}**

Specifies the fully qualified name of a Java class that is used to redirect output from this JVM. The class that you specify using the USEROUTPUTCLASS option extends `java.io.OutputStream`, and it handles all `System.err` and `System.out` requests from Java programs running in the JVM. CICS also uses this class to handle internal messages from the JVM, and event messages that

are directed to the destination specified in the `ibm.jvm.events.output` system property in the JVM properties file for the JVM (depending on the setting for that system property).

If you do not specify the `USEROUTPUTCLASS` option in a JVM profile, or if you specify it as null, the HFS files named by the `STDOUT` and `STDERR` options in the profile are used for output from the JVM. If you specify the `USEROUTPUTCLASS` option in a JVM profile, the HFS files named by the `STDOUT` and `STDERR` options in the profile could be used if the class named by the `USEROUTPUTCLASS` option is unable to write data to its intended destination.

This option is irrelevant in the JVM profile for the master JVM that initializes the shared class cache, because the output redirection class will never be invoked by the activities of the master JVM.

CICS supplies the sample class `com.ibm.cics.samples.SJMergedStream`, which produces merged log files containing the output from all JVMs with identifying headers, and the alternative sample class `com.ibm.cics.samples.SJTaskStream`, which produces HFS files containing the output from a single task. “Redirecting JVM output” in *Java Applications in CICS* describes the behaviour of these classes. The classes are shipped as a middleware class file `dfjoutput.jar`, which is in the directory `/usr/lpp/cicsts/cicsts31/lib`, where `cicsts31` is a user-defined value that you chose for the `CICS_DIRECTORY` variable used by the `DFHJVMJ` job during CICS installation. The source for the classes is also provided as samples, so you can modify the classes as you want, or write your own classes based on the samples.

The class that you are using must be present in a directory on the trusted middleware class path used by the JVM (to which you can add paths by using the `TMPREFIX` or `TMSUFFIX` option in the JVM profile). If you are using your own class in place of the supplied sample class, any associated native code for your class must be present on the library path used by the JVM, and must be explicitly loaded using the `System.loadLibrary()` call, either at class load time via a static initializer, or in the class constructor. (This avoids the need to include `doPrivileged()` blocks around the `loadLibrary` call when you are running with Java security active.) Note that if the JVM is to use the shared class cache (if `CLASSCACHE=YES` is specified in the JVM profile), you will need to include the class and any associated native code in the trusted middleware class path and library path that are specified in the JVM profile for the master JVM that initializes the shared class cache, rather than those specified in the JVM profile for the JVM itself. The CICS-supplied sample JVM profile for the master JVM is `DFHJVMCC`, and the JVM properties file that it references is `dfjvmcc.props`.

The Java programs that will run in JVMs that use the `USEROUTPUTCLASS` option should include appropriate exception handling to deal with the exceptions that might be thrown by the class named on the `USEROUTPUTCLASS` option. The CICS-supplied sample classes handle all exceptions internally, so they do not return any errors to the calling program.

Specifying the `USEROUTPUTCLASS` option has a negative effect on the performance of JVMs. For best performance in a production environment, you should not use this option. However, it can be useful to specify the `USEROUTPUTCLASS` option during application development. This is because if you do not specify `USEROUTPUTCLASS`, the output from all JVMs goes to the files specified by the `STDOUT` and `STDERR` options in the JVM profile. You can generate separate versions of these files for each JVM, but these files can only be differentiated by the `applid` of the CICS region and the time of

generation. The USEROUTPUTCLASS option enables developers using the same CICS region to separate out their own JVM output, and direct it to an identifiable destination of their choice.

**WORK\_DIR**={. *|directory\_name*}

Specifies the working directory on HFS that the CICS region uses for Java-related activities. The CICS JVM interface uses this directory when creating the `stdin`, `stdout` and `stderr` files. A period (`.`) is defined in the CICS-supplied JVM profiles, which means that the home directory of the CICS region user ID (that is, the HFS directory `/u/CICS region userid`) is to be used as the working directory. You can create this directory when you follow the process described in “Giving CICS regions access to UNIX system services and HFS directories and files” in *Java Applications in CICS*. If the CICS region user ID does not have this home directory, or if `WORK_DIR` is omitted altogether, `/tmp` is used as the HFS directory name.

You can create a subdirectory within this HFS directory to hold the output files, by specifying the subdirectory name after the period. For example, if you specify:

```
WORK_DIR=./javaoutput
```

then the output files from all the JVMs in that CICS region are created in the subdirectory `javaoutput` in the home directory of the CICS region `userid`.

If you do not want to use this home directory as the working directory for Java-related activities, or if your CICS regions are sharing the same z/OS user identifier (UID) and so have the same home directory, you can create a different working directory for each CICS region. You can do this by specifying a directory name that uses the `&APPLID` symbol, for which CICS substitutes the actual CICS region `APPLID`. This enables you to have a unique working directory for each region, even if all the CICS regions share the same set of JVM profiles. For example, if you specify:

```
WORK_DIR=/u/&APPLID;/javaoutput
```

then each CICS region using that JVM profile will have its own working directory. If you do this, ensure that you have created all the relevant directories on HFS and given the CICS regions read, write and execute access to them, as described in “Giving CICS regions access to UNIX system services and HFS directories and files” in *Java Applications in CICS*.

It is also possible to specify a fixed file name for the working directory, again ensuring that you have created the relevant directory on HFS and given the CICS regions the correct access. Bear in mind that when you use a fixed file name for the working directory, the output files from all the JVMs in the CICS regions that share the JVM profile are created in that directory. If you have also used fixed file names for your output files, the output from all the JVMs in those CICS regions will be appended to the same HFS files. If you want to avoid this, you can use **-generate** with the `STDERR` and `STDOUT` options to generate individual HFS files for each JVM in each CICS region. However, note that the use of the **-generate** option is not recommended in a production environment, because it can be detrimental to the performance of your JVMs.

You are recommended not to define your working directories within the CICS directory on HFS that contains the CICS-supplied Java code (the directory specified by the `CICS_DIRECTORY` option in the JVM profile, which by default is `/usr/lpp/cicsts/cicsts31/`).

You can also use the option `USEROUTPUTCLASS` to name a Java class that intercepts, redirects and formats the `stderr` and `stdout` output from a JVM. The CICS-supplied sample classes for output redirection use the directory specified by `WORK_DIR` in some circumstances.

## Java standard options

The following Java standard options are passed by CICS to the JVM when the JVM is being launched:

### `CLASSPATH=class_pathnames`

Specifies the directory paths to be searched by the JVM for nonshareable application classes and resources—that is, application classes that you do not want to be shared by other JVMs or across JVM resets. If the JVM is defined as resettable, classes on this class path are discarded when the JVM is reset, and reloaded from HFS files each time the JVM is reused. If the JVM is defined as a continuous JVM, however, nonshareable application classes are kept intact from one JVM reuse to the next. If the JVM is defined as a worker JVM, nonshareable application classes are not placed in the shared class cache. This class path is known as the standard class path. In CICS, this option in the JVM profile is used instead of the `java.class.path` option in the JVM properties file. This option is ignored if it is specified in the JVM profile used for the master JVM that initializes the shared class cache, because applications cannot run in the master JVM.

If the JVM is a resettable JVM, which is the case if the option `REUSE=RESET` (or the older option `Xresettable=YES`) is specified in the JVM profile, you should avoid using the `CLASSPATH` option in a production environment, because it is detrimental to performance. If the JVM is a worker JVM (either resettable or continuous), which is the case if the option `CLASSCACHE=YES` is specified in the JVM profile, using the `CLASSPATH` option increases storage usage, because the classes are stored in every JVM instead of just in the master JVM. Instead, use the `ibm.jvm.shareable.application.class.path` system property. For standalone JVMs with `CLASSCACHE=NO` in their JVM profile, this system property is specified in the JVM properties file for the JVM, but for worker JVMs with `CLASSCACHE=YES` in their JVM profile, it is specified in the JVM properties file for the master JVM that initializes the shared class cache.

You can specify directory paths on separate lines by using a `\` (backslash) at the end of each line that is to be continued.

If any of your Java application programs are built as a package (that is, compiled using a Java package statement), do *not* include the package name as part of the path name on `CLASSPATH`.

For example, the source of the CICS `HelloCICSWorld` sample program begins with:

```
package examples.HelloWorld;
```

In this case, the package name should be included in the class name in the CICS program resource definition; for example, as `JVMCLASS(examples.HelloWorld.HelloCICSWorld)`.

When you create the sample program using the supplied `HelloCICSWorld.makefile`, it is installed in the `/examples/HelloWorld/` subdirectories. For example, if you build the samples in the default directories created when you installed CICS TS, the full path is

```
/usr/lpp/cicsts/cicsts31/samples/dfjcics/examples/HelloWorld/
```

The correct CLASSPATH to enable the JVM to find this Java program is  
CLASSPATH=/usr/lpp/cicsts/cicsts31/samples/dfjcics

omitting the package name.

**DISABLEASSERTIONS={ALL|*packageName*...|*className*}**

Disables assertion checking for the specified code at run time. For details of the syntax, and the rules for using this option with applications that run in worker JVMs, see the ENABLEASSERTIONS option.

**ENABLEASSERTIONS={ALL|*packageName*...|*className*}**

Enables assertion checking for the specified code at run time. An assertion is a statement in the Java programming language that enables you to test your assumptions about your program. You can specify that assertions should be enabled in a package and in any subpackages that it has (the ... after the package name ensures that any subpackages are included), or in an individual class. Specifying ALL enables assertions in all classes except system classes.

You can specify more than one option relating to assertions in the JVM profile. For example, you could use the ENABLEASSERTIONS option to specify a package for which you want to enable assertions, and then use the DISABLEASSERTIONS option to disable assertions for a particular class within that package. However, note that CICS does not support the use of multiple instances of the *same* option relating to assertions. For example, you cannot specify the DISABLEASSERTIONS option twice in order to disable assertions for two individual classes.

If you are using assertions with applications that run in worker JVMs (JVMs that share the class cache), the following rules apply:

- Specifying ENABLEASSERTIONS=ALL or DISABLEASSERTIONS=ALL in a *master* JVM enables or disables assertions for all the classes (except for system classes) that are in the shared class cache—that is, the middleware classes and the shareable application classes. Specifying ENABLEASSERTIONS=ALL or DISABLEASSERTIONS=ALL in a *worker* JVM enables or disables assertions for all the classes that are in the worker JVM—that is, any nonshareable application classes that you placed on the standard class path (specified by the CLASSPATH option in the JVM profile).
- If you are using the ENABLEASSERTIONS or DISABLEASSERTIONS option to enable or disable assertions for a package or an individual class, place the option in the JVM profile that contains the class path on which the package or class is named. So for middleware classes and shareable application classes, or the packages containing them, place the option in the JVM profile for the master JVM. For nonshareable application classes that you placed on the standard class path (specified by the CLASSPATH option in the JVM profile), place the option in the JVM profile for the worker JVM.

You can find more information about programming with assertions, and about enabling and disabling assertions, at <http://java.sun.com/j2se/1.4.1/docs/guide/lang/assert.html>.

**SYSTEMASSERTIONS={ON|OFF}**

Enables or disables assertion checking for all system classes in the JVM. To enable or disable assertion checking for individual system classes, use the ENABLEASSERTIONS and DISABLEASSERTIONS options as you would for any other classes.

The SYSTEMASSERTIONS option can be specified in a master JVM, but it causes an error in a worker JVM (because the worker JVM does not have its own system heap). If you want to enable or disable assertions for system

classes used by an application that runs in a worker JVM, specify the `SYSTEMASSERTIONS` option in the JVM profile for the master JVM.

**VERBOSE**={NO|[`class`][,`gc`][,`jni`]}

Indicates whether or not the JVM should issue a message containing information about its activities. You can specify any or all of the three available options, or none. The options are:

**NO** Omit the verbose option so that the JVM does not report any of the information messages described by the following options.

**class** Specifies that the JVM is to report information about each class it loads. This equates to the standard JVM launcher option `-verbose` (or `-verbose:class`) defined in the persistent reusable JVM Java 2 specification.

**gc** Specifies that the JVM should issue a message to report each garbage collection event. This equates to the standard JVM launcher option `-verbose:gc` defined in the persistent reusable JVM specification.

**jni** Specifies that the JVM should issue a message each time it performs one of the following Java native interface (JNI) operations:

- Dynamic link of a native method
- Registers a native method
- Loads a native library.

This equates to the standard JVM launcher option `-verbose:jni` defined in the persistent reusable JVM specification.

Coding examples:

```
verbose=class
verbose=class,jni
verbose=gc
```

## Java nonstandard options

The following are the nonstandard options that are supported by the IBM persistent reusable JVM.

**Note:** Specify storage sizes in multiples of 1024 bytes. Use the letter K to indicate kilobytes, the letter M to indicate megabytes, and the letter G to indicate gigabytes. For example, to specify 6291456 bytes as the initial size of the middleware heap, code `Xms` in one of the following ways:

```
Xms=6291456
Xms=6144K
Xms=6M
```

**Xcheck**={NO|[`jni`][,`nabounds`]}

Specifies whether or not you want the JVM to perform additional checks. There are two options, and you can specify either of these or both.

**jni** Performs additional checks for JNI functions

**nabounds**

Performs a native-array bounds check for JNI functions.

If you omit the `jni` option, also omit the comma in front of `nabounds`.

**Xdebug={NO|YES}**

Specifies whether or not debugging support is to be enabled in the JVM. See also the Xrunjdpw debug invocation option.

For more information about using a debugger with the JVM, see “Using a debugger with the JVM” in *Java Applications in CICS*. See also the Java Platform Debugger Architecture (JPDA) description at <http://java.sun.com/products/jpda/doc/>.

To ensure clean termination of the debug session, specify REUSE=NO (or the older option Xresettable=NO) when debugging support is enabled.

In JVM profiles where you have specified **CLASSCACHE=YES** (that is, the JVM profiles for worker JVMs), this option is ignored. It is also ignored in the JVM profile for the master JVM that initializes the shared class cache. Worker JVMs and the master JVM cannot be configured for debug.

**Xinitacsh={128KB|size}**

Specifies the initial size of the application-class system heap for resettable JVMs. This option is ignored for continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), because these types of JVM do not have an application-class system heap. It is also ignored for worker JVMs (those with CLASSCACHE=YES in their JVM profile), because worker JVMs use the master JVM's system heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 128KB.

**Xinitsh={128KB|size}**

Specifies the initial system heap size. The JVM does not enforce any maximum heap size. This option is ignored for worker JVMs (those with CLASSCACHE=YES in their JVM profile), because worker JVMs use the master JVM's system heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 128KB.

**Xinitth={500KB|size}**

Specifies the initial transient heap size. This option only applies to resettable JVMs. It is ignored for:

- Continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), because these types of JVM do not have a transient heap.
- The master JVM that initializes the shared class cache.

The storage for the transient heap is allocated from the high-end of the storage area allocated by the Xmx option, and is allowed to expand down until it meets the boundary of the middleware heap (specified by the Xms option).

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 500KB. If Xinitth is not specified and Xms (which sets the initial size of the middleware heap) is specified, then Xinitth is set to half of Xms.

**Xmaxe={0|size}**

Specifies the maximum heap expansion size for the middleware and transient heaps. Continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), and the master JVM that initializes the shared class cache, have no transient heap, so for those JVMs the value only applies to the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 0 (there is no maximum heap expansion size).

**Xmaxf={0.6|number}**

Specifies the maximum percentage of free space for the middleware heap, as a number between 0 and 1. This value applies only to the middleware heap and not to the transient heap (if the JVM has one). The default is 0.6 (that is, 60%). If a value less than 1 is specified and more free space is available than the value specified, the middleware heap is shrunk. A value of 0 allows no free space in the heap, and so heap contraction is a constant activity. A value of 1 means that the heap never contracts.

**Xmine={1MB|size}**

Specifies the minimum heap expansion size for the middleware and transient heaps. Continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), and the master JVM that initializes the shared class cache, have no transient heap, so for those JVMs the value only applies to the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 1MB.

**Xminf={0.3|number}**

Specifies the minimum percentage of free space for the middleware and transient heaps. Continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), and the master JVM that initializes the shared class cache, have no transient heap, so for those JVMs the value only applies to the middleware heap. As for the Xmaxf option, this is specified as a number between 0 and 1. The default is 0.3 (that is, 30%). If the free space falls below this amount, the heap grows.

**Xms={500KB|size}**

Specifies the initial size of the middleware heap. The storage for the middleware heap is allocated from the low-end of the storage area allocated by the Xmx option, and is allowed to expand up until it meets the boundary of the transient heap (specified by the Xinitth option).

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 500KB.

**Xmx={64MB|size}**

Specifies the maximum total size of the middleware and transient heaps (that is, the size of the non-system heap). Note that this fixed amount of storage is allocated by the JVM during JVM initialization. The middleware heap grows from the bottom of this region, and the transient heap grows from the top of the region.

Continuous and single-use JVMs (those with REUSE=YES, REUSE=NO or Xresettable=NO in their JVM profile), and the master JVM that initializes the shared class cache, do not have a transient heap, so for these JVMs, the value of Xmx sets the maximum size for the middleware heap.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default is 64MB.

**Xnoclassgc={NO|YES}**

Specifies whether or not you want CICS to include the Xnoclassgc option on JVM launch command.

**NO** Omitting the option means that the JVM performs class garbage collection.

**YES** The option is specified on the JVM launch command, which means that class garbage collection is not performed.

**Xoss={400KB|size}**

Specifies the maximum size of the Java code stack for each thread. Each Java thread has two stacks: one for Java code and one for C code. This option sets the maximum stack size that can be used by Java code in a thread.

This option does not affect the allocation of storage, but the JVM uses it to check whether the specified limit has been exceeded, in which case the JVM throws a Java StackOverflowException.

Specify *size* as a number of bytes, kilobytes, or megabytes (see “Specifying storage sizes” on page 305). The default size is 400KB.

**Xquickstart**

See the Xservice="-Xquickstart" option.

**Xresettable={YES|NO}**

This option is a deprecated method of specifying the level of reusability for a JVM. “How JVMs are reused and reset” in *Java Applications in CICS* explains the three levels of reusability that a JVM can have. For migration purposes, the Xresettable option is still a valid way to create a resettable JVM (which is specified by Xresettable=YES) or a single-use JVM (which is specified by Xresettable=NO), but it cannot be used to create a continuous JVM. You should now use the REUSE option in the JVM profile to specify the level of reusability for the JVM. The CICS-supplied sample JVM profiles use the REUSE option instead of the Xresettable option.

When you use the REUSE option, it is advisable to remove the Xresettable option from the JVM profile, although if the options conflict the REUSE option overrides the Xresettable option. An information message is issued if this happens.

**Xrs={NO|YES}**

Specifies whether or not you want CICS to include the Xrs option on the JVM launch command. Xrs reduces the use of operating system signals by the JVM.

**Xrundllname=(suboption=value,suboption=value...)**

Loads the specified dynamic link library (DLL) and passes it the list of options. This option is typically specified as a list of sub-options of the form <suboption=value>, with each sub-option separated by a comma. You can load as many DLLs as you want, by repeating the Xrundllname option. The DLLs that you specify using the Xrundllname option must be present in directories on the library path for the JVM, which is specified by the LIBPATH option in the JVM profile.

Some useful DLLs that you can load using this option are:

- **hprof**, a DLL that drives JVMPi to perform CPU, heap, or monitor profiling. To load this DLL, code

```
Xrunhprof=(suboption=value,suboption=value...)
```

For more information about hprof, see <http://java.sun.com/j2se/1.4/docs/guide/jvmpi/>

- **jdwp**, a DLL that drives JVMDI to perform debugging. To load this DLL, code

```
Xrunjdwp=(suboption=value,suboption=value...)
```

This option loads the JPDA reference implementation in-process debugging libraries and passes any `-Xrunjdw` sub-options specified. This library resides in the target VM and uses Java virtual machine debug interface (JVMDI) and the Java native interface (JNI) to interact with it. It uses a transport and the Java Debug Wire Protocol (JDWP) to communicate with a separate debugger application. For more information, see the Java Debugger (JDB) description in the Java 2 specification at <http://java.sun.com/products/jpda/doc/>

CICS uses the `Xrundllname` option to load `dfhapjvmt`, which drives the JVMRAS interface to perform tracing. CICS specifies this option automatically when it is required.

#  
#  
#  
#  
#  
#  
#  
#  
#  
#

#### **Xservice="-Xquickstart"**

Depending on your application, specifying this option in a JVM profile can provide some reduction in the startup time for those JVMs. The `Xservice="-Xquickstart"` option causes the just-in-time (JIT) compiler function of the JVM to omit certain optimizations when a method is first compiled. If the method is used frequently enough, the JIT compiler compiles it again using all the optimizations. This technique improves startup time for the JVM, but performance can be degraded on subsequent reuses of the JVM, because the code is not fully optimized or because of the time required for the second compile.

#### **Xss={500KB|size}**

Specifies the maximum size of the native code stack for each thread. Each Java thread has two stacks: one for Java code and one for C code. This option sets the maximum stack size that can be used by C code in a thread.

This option does not affect the allocation of storage, but the JVM uses it to check whether the specified limit has been exceeded, in which case the JVM throws a `Java StackOverflowException`.

Specify *size* as a number of bytes, kilobytes, or megabytes (see "Specifying storage sizes" on page 305). The default is 500KB. At least 1KB must be specified.

#### **Xverify={remote|all|none}**

Specifies the level of verification you want the JVM to perform on classes to be loaded. When class files are loaded (possibly over the network or from an untrusted source) into a JVM, there is no way of telling how their byte codes were generated.

The options are:

##### **remote**

Verify only those classes that are loaded over the network. The default verification setting means that anything installed locally is not verified. This option equates to the Java command line option `Xverify:remote`.

##### **all**

Verify everything. This option equates to the Java command line option `Xverify:all`.

##### **none**

Do not perform any verification. This option equates to the Java command line option `Xverify:none`.

The CICS-supplied sample JVM profiles specify `Xverify:none`.

For worker JVMs that use the shared class cache (those with `CLASSCACHE=YES` in their JVM profile), this option only operates on classes

that are on the standard class path (specified by the CLASSPATH option in the JVM profile). Other classes, on the shareable application class path or the middleware class path, are verified according to the verification setting specified for the master JVM that initializes the shared class cache.

---

## System properties for JVMs

"Setting up JVM profiles and JVM properties files" in *Java Applications in CICS* tells you how to set up suitable JVM profiles and JVM properties files to meet the needs of your applications.

System properties are key name and value pairs that contain basic information about the JVM and its environment, such as the operating system in which the application is running. The system properties are initialized when the JVM is first created. Some are set by the JVM automatically when it is initialized, and others you can specify in a JVM properties file, which is referenced by the JVMPROPS option in the JVM profile (see "Options in JVM profiles" on page 290).

CICS passes all the system properties in a JVM properties file to the JVM unchanged. However, you should bear in mind that only the system properties described in the CICS documentation are supported by CICS, although the JVM can support a much wider range of system properties. This topic documents the system properties that are particularly relevant for JVMs in a CICS environment, including some which are defined by CICS. *Persistent Reusable Java Virtual Machine User's Guide*, SC34-6201, lists command-line options, JVM options and system properties that are used in a persistent reusable JVM in a z/OS environment, some of which are provided in a different format by options in the JVM profile in CICS. The *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/](http://www.ibm.com/developerworks/java/jdk/diagnosis/), documents further system properties that are used for JVM trace and problem determination. The Java class libraries include other system properties, and applications might have their own system properties. There is no central repository of all system properties for the JVM.

Specify each system property according to the coding rules described in "Rules for coding JVM profiles and JVM properties files" on page 287. Remember that all parameter keywords and operands specified in a JVM profile or JVM properties file are case-sensitive, and must be specified exactly as shown in the following sections.

Property values are passed to the JVM for the construction of system properties as if specified by a -D option in a java command. For example, the JVM properties file could set the java.security.policy property by specifying:

```
java.security.policy=  
file:/usr/lpp/cicsts/cicsts31/lib/security/dfjejbp1.policy
```

(where *cicsts31* is a user-defined value that you specify during the installation of CICS TS). CICS uses this information to create a Java launcher option as follows:

```
-Djava.security.policy=  
file:/usr/lpp/cicsts/cicsts31/lib/security/dfjejbp1.policy
```

**JVM properties files for worker JVMs (referenced by JVM profiles that specify CLASSCACHE=YES):**

As described in “Options in JVM profiles” on page 290, if you specify **CLASSCACHE=YES** in a JVM profile, making the JVM a worker JVM that uses the shared class cache, certain system properties in the JVM properties file referenced by the `JVMPROPS` option on the JVM profile are treated differently. The affected system properties are **`ibm.jvm.shareable.application.class.path`** and **`java.compiler`**. The descriptions for these system properties tell you how to handle them for worker JVMs.

### **JVM properties file for the master JVM that initializes the shared class cache:**

As described in “Options in JVM profiles” on page 290, the master JVM that initializes the shared class cache is not used to run Java applications. Because of this, most of the system properties that would normally be specified for a JVM are not needed for the master JVM. The CICS-supplied sample JVM profile for the master JVM is `DFHJVMCC`, and the JVM properties file that it references is `dfjjvmcc.props`.

In the JVM properties file for the master JVM, the only system property that needs to be specified is `ibm.jvm.shareable.application.class.path`, which should specify the paths to the shareable application classes for all the applications that will run in worker JVMs that use the shared class cache.

The `java.compiler` system property can be included in the JVM properties file for the master JVM, or allowed to default to `jitc`. Also, if you want to activate tracing for the master JVM, you can specify the `ibm.dg.trc.external` system property, which should only be used with care. These system properties are not specified in the CICS-supplied sample JVM properties file `dfjjvmcc.props`.

The remaining system properties, other than those that are set automatically by the IBM JVM, are irrelevant for a master JVM. They do not cause an error if they are specified in the JVM properties file for the master JVM, but you should not include any of them unless you do so under the direction of Level 2 support.

### **Customizing `dfjjvmcd.props`:**

As described in “Options in JVM profiles” on page 290, the JVM profile `DFHJVMCD`, and its associated JVM properties file, `dfjjvmcd.props`, are reserved for use by CICS-supplied system programs, in particular the default request processor program `DFJIIRP` (used by the CICS-supplied `CIRP` request processor transaction).

Only make the changes to `dfjjvmcd.props` that are necessary to run applications, as follows:

- Add to the shareable application class path (the `ibm.jvm.shareable.application.class.path` system property) any classes, such as classes for utilities, that are required by your enterprise beans but are *not* included in the deployed JAR files for the enterprise beans.
- Specify the system properties necessary to configure your JNDI nameserver (the `com.ibm.cics.ejs.nameserver` system property, and further system properties if you are using an LDAP nameserver).
- Enable the Java 2 security policy mechanism (the `java.security.policy` system property) if required to do so by your installation.

Do not make any other changes to `dfjjvmcd.props`.

### Security caution

You should ensure that the JVM properties files are secure, with update authority restricted to system administrators. This is because the JVM properties files are typically used to define sensitive JVM configuration options, such as the security policy file and the trusted middleware class path.

In particular, if you specify that a secure LDAP server is to be used, by coding `java.naming.security.authentication` in the JVM properties files, you also need to specify `java.naming.security.principal` and `java.naming.security.credentials`. These properties hold the UserID and password that CICS requires to access the secure LDAP service, so you need to give particular attention to the access controls in force at your installation for the JVM properties files, and any other copies of this information that you have.

## Standard system properties

The standard Java system properties, common to all JVMs, are as follows:

### `file.separator=`

The file separator character. By default, this is set by the JVM as the `/` symbol. Do not change this value.

### `java.class.path=`

The standard class path, for nonshareable application classes. **Do not use this system property in a JVM properties file for CICS.** In CICS, the standard class path is specified instead by the `CLASSPATH` option in the JVM profile.

### `java.class.version=`

The Java class version number. This is set by the IBM JVM as 048. Do not change this value.

### `java.compiler={j|tc | NONE }`

The Java just-in-time (JIT) compiler. This is set by default as `java.compiler=j|tc`, which means that after a pre-determined number of invocations of a Java method, the JIT compiler is invoked to compile the bytecode into z/OS machine code.

This system property can be specified in the JVM properties file for a master JVM.

In JVM properties files referenced by JVM profiles where you have specified **CLASSCACHE=YES** (that is, the JVM properties files for worker JVMs), this system property must either:

- Specify `java.compiler=NONE` (the word `NONE` must be specified in upper case), to turn off the Java just-in-time (JIT) compiler for the worker JVM.
- or
- Specify the same value that is specified in the JVM properties file for the master JVM.

The activities of the Java just-in-time (JIT) compiler can interfere with the logging of unresetable events, reset trace events and cross-heap events. During the development process, specify `java.compiler=NONE` to turn off the JIT compiler for the JVM. Remember to turn the JIT compiler back on when you have finished investigating unresetable events, reset trace events and cross-heap events in your application.

**java.home=**

The Java installation directory. Do not change this value.

**java.naming.provider.url=**

Identifies a COS Naming Service. This property continues to be supported only for compatibility with CICS TS 2.1. You are recommended to use the `com.ibm.cics.ejs.nameserver` property (see “`com.ibm.cics.ejs.nameserver`” on page 316) instead.

- If you use an LDAP name server you *must* use the `com.ibm.cics.ejs.nameserver` property to identify the name server.
- If you use a COS Naming Service, you are *recommended* to use the `com.ibm.cics.ejs.nameserver` property, rather than `java.naming.provider.url`. This avoids the possibility of providing contradictory information to CICS.

If you *do* use the `java.naming.provider.url` property, this sets the protocol, host name, and port number of the COS Naming Service that CICS is to use. For example,

```
java.naming.provider.url=IIOP://servername.hursley.ibm.com:900
```

For special considerations that apply if you are using the COS Naming Directory Server supplied with WebSphere Application Server Version 5 or later, see “`com.ibm.cics.ejs.nameserver`” on page 316.

You are strongly recommended to avoid using *both* the `java.naming.provider.url` and the `com.ibm.cics.ejs.nameserver` property in the same JVM properties file, because:

- If the latter specifies an LDAP name server while the former specifies a COS Naming Service, results are unpredictable.
- If both properties specify different COS Naming services, the one defined by the `java.naming.provider.url` property takes precedence.

**java.naming.security.authentication=**

specifies the type of security authentication in use for naming operations. This property may be required if you are using an LDAP name server.

CICS needs to have write access into the LDAP namespace. If the LDAP service is set up securely, these three properties—authentication, credentials and principal—are required. If the LDAP service is set up so that any user can write to it, these three are not needed. Your LDAP administrator can tell you whether or not you need to include these properties in your JVM properties file.

*Simple* is the only value for this property that is supported by CICS. Specifying `java.naming.security.authentication=simple` indicates that the LDAP name server is running in secure mode.

**Important**

If you do specify this property, you have also to specify `java.naming.security.principal` and `java.naming.security.credentials`.

Because these properties specify the user ID and password that CICS requires to access the secure LDAP service, you need to give particular attention to the file permissions controlling access to the JVM properties file and any other copies of this information that you have.

**java.naming.security.credentials=**

specifies the password required for the **principal** (see "java.naming.security.principal") to access to the LDAP name server.

This property is required if you specified `java.naming.security.authentication=simple`. Your LDAP administrator provides the value that you should specify, for example, `java.naming.security.credentials=secret`.

**java.naming.security.principal=**

specifies the **principal** required for access to the LDAP name server.

This property is required if you specified `java.naming.security.authentication=simple`. Your LDAP administrator provides the value that you should specify, for example, `java.naming.security.principal=cn=CICSUser,c=uk`.

**Notes:**

1. The principal/credentials of `cn=CICSAdmin,c=uk/secret` is the default which applies if the administrator makes no changes to the CICS scripts used when building the system namespace.
2. For more information about **principals**, see *Java Applications in CICS*. You may find it helpful to think of **credentials** as 'passwords', and **principals** as 'userIDs'.

**java.security.manager={default | "" | *other\_security\_manager*}**

This system property indicates the Java security manager to be enabled for the JVM. To enable the default Java 2 security manager, include this system property in one of the following formats:

```
java.security.manager=default
```

or

```
java.security.manager=""
```

or

```
java.security.manager=
```

All these statements have the effect of enabling the default security manager. If you do not include the `java.security.manager` system property in your JVM properties file, then the JVM runs without Java 2 security enabled. If you need to disable Java 2 security for a JVM, comment out this system property.

**java.security.policy=**

This system property describes the location of additional policy files that you want the security manager to use to determine the security policy for the JVM. A default policy file is provided with the JVM in `/usr/lpp/java142/J1.4/lib/security/java.policy`, where the `java142/J1.4` subdirectory names are the default values when you install the IBM Developer Kit for OS/390, Java 2 Technology Edition. The default security manager always uses this default policy file to determine the security policy for the JVM, and you can use the `java.security.policy` system property to specify any additional policy files that you want the security manager to take into account as well as the default policy file.

To enable CICS Java applications and enterprise beans to run successfully when Java 2 security is active, you need to specify, as a minimum, an additional policy file that gives CICS the permissions it needs to run the

enterprise beans container, and gives applications the permissions outlined in the *Enterprise JavaBeans* specification, Version 1. The CICS-supplied enterprise beans policy file, `dfjejbpl.policy`, contains the permissions that you need for this purpose. To specify this policy file, include the system property:

```
java.security.policy=/usr/lpp/cicsts/cicsts31/lib/security/dfjejbpl.policy
```

where `cicsts31` is your chosen value for the USSDIR installation parameter that you defined when you installed CICS TS.

“Managing security for enterprise beans” in *Java Applications in CICS* has more information about specifying security policy files, and about `dfjejbpl.policy`.

**java.vendor=**

Java vendor-specific string. This is set by the IBM JVM as “IBM Corporation”.

**java.vendor.url=**

The Web address of the vendor's home page. Set by the IBM JVM as “`http://www.ibm.com/`”

**java.version=**

The Java version number of the Java run-time environment. This is set by the IBM JVM as “1.4.2”.

**jdbc.drivers=**

Specifies one or more JDBC drivers. Setting the driver names as a system property is an alternative to the Java application itself loading the drivers using the `Class.forName("driver_name");` command. Separate each driver name in a list by a : (colon). An example of this system property is included within comments in the CICS-supplied system property file `dfjjvmpr.props`.

To specify the DB2-supplied JDBC drivers, set the system property as:

```
jdbc.drivers=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver
```

This is a common name that works for all levels of JDBC driver supplied by DB2, including the DB2 Universal JDBC Driver.

“Using JDBC and SQLJ to access DB2 data from Java programs and enterprise beans written for CICS” in the *CICS DB2 Guide* has more information about using JDBC.

**line.separator=**

The line separator character. This is set by default as `\n`.

**os.arch=**

The operating system architecture. This is set by the IBM JVM as “390”

**os.name=**

The operating system name. This is set by the IBM JVM as “z/OS”. Do not change this value.

**path.separator=**

The character used to separate paths in a multipart class path. By default, this is set as a : (colon). Do not change this value.

**user.dir=**

The user's current working directory. In this case, the user is identified by the CICS region's userid. Do not change this value.

**user.home=**

The user's home directory. In this case, the user is identified by the CICS region's userid. Do not change this value.

**user.name=**

The user's account name. In this case, this is set by the JVM to the account name under which the CICS job is running. Do not change this value.

## System properties specific to the IBM persistent reusable JVM

In addition to the basic system properties, shown in "Standard system properties" on page 312, which are common to all JVMs, there are some that are unique to the IBM persistent reusable JVM. These are listed below.

**com.ibm.cics.datasource.path=**

specifies the name and subContext of a CICS-compatible DataSource that you have deployed to generate JDBC connections for Java applications in CICS that access DB2. The *CICS DB2 Guide* has more information about this.

**com.ibm.cics.ejs.nameserver=**

specifies the URL and TCP/IP port number of the name server that you use for JNDI references. For example:

- For an LDAP name server, specify something like:

```
com.ibm.cics.ejs.nameserver=ldap://myldserv.hursley.ibm.com:389
```

where myldserv.hursley.ibm.com is the URL of the name server and 389 is the port number on which it is configured to listen. Your LDAP administrator can supply the correct URL and port number.

- For a standard COS Naming Directory Server, specify something like:

```
com.ibm.cics.ejs.nameserver=iio://mycsserv.hursley.ibm.com:900
```

The relevant administrator in your organisation can supply the correct name and port number.

If you are using the COS Naming Directory Server supplied with WebSphere Application Server Version 5 or later, you should specify:

```
com.ibm.cics.ejs.nameserver=iio://mycsserv.hursley.ibm.com:2809/domain/legacyRoot
```

This is because, in WebSphere Application Server:

- The default TCP/IP port used by the COS Naming Directory Server is 2809.
- CICS objects must be published to a specially-architected location (in the WebSphere® naming structure) called "domain/legacyRoot". (CICS publishes objects to a context defined by the JNDIPREFIX option of the CORBASERVER definition, where the JNDI prefix is a relative path.) If you do not specify the /domain/legacyRoot path from the root node of the name space, CICS tries to publish objects to the JNDI prefix location relative to the root node itself. With the COS Naming Directory Server supplied with WebSphere Application Server, this fails.

An example of this statement is included in the CICS-supplied sample JVM properties file, dfjjvmpr.props.

**Note:** If you are using a COS naming service, and you have chosen to specify it in java.naming.provider.url, do not specify it again here.

**com.ibm.cics.ejs.loadjndiproperties=**

You can set up a file called jndi.properties to contain JNDI nameserver configuration properties that are common across a set of CICS regions. By default, CICS does not attempt to locate a jndi.properties file. Include the following system property to cause CICS to load jndi.properties for this JVM:

```
com.ibm.cics.ejs.loadjndiproperties=true
```

Place the directory containing the `jndi.properties` file on either the shareable application class path (in the JVM properties file) or the trusted middleware class path (in the JVM profile), in all the relevant JVM profiles or JVM properties files, for all the regions that you want to share the same nameserver settings.

#  
#  
#  
#  
#

**com.ibm.cics.soap.validation.local.CCSID=**

specifies the local code page to use when validating SOAP messages if validation is enabled for a CICS WEBSERVICE resource. If a local CCSID is not specified then the default USS code page for your installation is assumed when validating the SOAP message.

**com.ibm.websphere.naming.jndicache.cacheobject={populated | none}**

Turns the JNDI cache on or off. The JNDI cache stores the results of JNDI lookups in local storage, so that, if an application does the same lookup twice (perhaps in different tasks), the results are already available. Note that the cache:

- Is JVM-specific. That is, there is a separate cache for each JVM.
- Only works with an IBM JNDI name server.
- Stores only object references (and not other things, such as DataSources).

**populated**      The JNDI cache is active.

**none**            The JNDI cache is not used.

**com.ibm.websphere.naming.jndicache.maxcachelife={20 |mins}**

Specifies, in minutes, the “time to live” of the JNDI cache. If the cache is accessed after this time is exceeded the entire cache is flushed of its contents.

See also the **com.ibm.websphere.naming.jndicache.cacheobject** property.

**com.ibm.ws.naming.ldap.containerdn=**

specifies the **Container Distinguished Name** for the LDAP name server. For example:

```
com.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=t1,o=WASNaming,c=us
```

An example of this statement is included in the CICS-supplied sample JVM properties file, `dfjjvmpr.props`. Your LDAP administrator can supply the correct value for your installation.

The **Container Distinguished Name** is the root of the system name space.

This property is not required if you specify a COS naming service.

**com.ibm.ws.naming.ldap.noderootrdn=**

specifies the **Noderoot Relative Distinguished Name** for the LDAP name server. For example:

```
com.ibm.ws.naming.ldap.noderootrdn=ibm-wsnName=legacyroot,  
ibm-wsnName=PLEX2,ibm-wsnName=domainRoots
```

Your LDAP administrator can supply the correct value.

An example of this statement is included in the CICS-supplied sample JVM properties file, `dfjjvmpr.props`.

This property is not required if you specify a COS naming service.

**ibm.dg.trc.external=**

Setting this system property sets trace options and enables tracing for the JVM. This enables you to trace a JVM during its whole lifetime, including start-up and reset as well as the periods when it is being used by a transaction. This system property has to be used with care, as JVM tracing can produce large amounts

of output in a very short time. If you only want to activate JVM trace for particular transactions that use the JVM, use the CICS-supplied transaction CETR, rather than this system property. “Controlling tracing for JVMs” in *Java Applications in CICS* has more information about this. “Defining tracing for JVMs” in the *CICS Problem Determination Guide* has information about the JVM trace options that you can set using this system property. There is further information about JVM trace and about problem determination for JVMs in the *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/](http://www.ibm.com/developerworks/java/jdk/diagnosis/).

This system property can be specified in the JVM properties file for a master JVM.

When CICS starts to use or re-use a JVM, it ensures that any trace options that you have set and activated using CETR are applied. Activating or deactivating a trace option using CETR overrides any setting for that trace option in the `ibm.dg.trc.external` system property. For example, a trace option that is activated in the system property, but deactivated using CETR, will be deactivated when CICS starts to use or re-use the JVM. If you use CETR to activate any trace options that are not referred to in the `ibm.dg.trc.external` system property, the trace options that you have specified in CETR are *added* to any trace options that you have set using the `ibm.dg.trc.external` system property. The trace output will then reflect all the trace options that you requested in both CETR and the system property.

When you activate JVM trace, the JVM trace appears as CICS trace point SJ 4D01. If the JVM trace facility fails, CICS issues the trace point SJ 4D00.

#### **`ibm.jvm.crossheap.events={on}`**

Setting this system property enables the logging of cross-heap references in a resettable JVM (that is, a JVM with the option **REUSE=RESET** specified in its JVM profile). To see this information, you also need to set the `ibm.jvm.events.output` system property to enable event logging.

This system property is not set in the CICS-supplied JVM properties files, so by default the function is not enabled. Note that when you include this system property in the JVM properties file with any value (“on” is one possibility), the function is enabled. To disable the function, you need to comment out or remove the system property; there is no value you can specify for the system property that disables the function.

Cross-heap references are references between the middleware heap and the transient heap in the JVM. When you set up the JVM as described here, cross-heap references are logged to the event output destination at the time that each reference is created. The log entry includes a full stack trace to identify the line of code that created the cross-heap reference.

Most of the cross-heap references that are logged will be removed before the JVM is reset, through the normal actions of the CICS and JVM code, and through any actions that your application takes for this purpose. However, if any cross-heap references are not removed before the JVM is reset, they cause the JVM to perform a trace-for-unresettability check. Any references that are found to be in live objects trigger unresettable events, which cause the JVM to be marked as unresettable and destroyed. Any references that are found to be in unreferenced middleware heap objects (garbage) are reported as reset trace events, which do not cause the JVM to be destroyed, but have still wasted processor time by causing the trace-for-unresettability check. You should

therefore ensure that all cross-heap references created by your applications are removed from the JVM before it is reset.

You can use the memory location listed for an unresettable event or a reset trace event to identify the cross-heap reference recorded in the event log which is responsible for triggering the event. You can then use the stack trace associated with the cross-heap reference to help you to fix the problem. You might have to perform compensatory actions in application code to cause a cross-heap reference to be removed, which could include closing files or streams, emptying collections, or other kinds of clean-up activity. If you cannot remove the cross-heap reference in application code, consider contacting your IBM support representative for further advice.

The activities of the Java just-in-time (JIT) compiler can interfere with the logging of cross-heap references. During the development process, specify the system property `java.compiler=NONE` in the JVM properties file to turn off the JIT compiler for the JVM. (The word `NONE` must be in upper case.) When you have finished investigating cross-heap references in your application, remember to turn the JIT compiler back on.

**`ibm.jvm.events.output={event.log | path/file_name | stderr | stdout}`**

Setting this system property enables event logging in a resettable JVM (that is, a JVM with the option **REUSE=RESET** specified in its JVM profile). You can store the text records describing the events in a HFS file, or in the `stderr` or `stdout` files for the JVM. You can specify only the name for the HFS file, in which case the file is created in the directory specified by the `WORK_DIR` option in the JVM profile. Alternatively, you can specify a full HFS path and file name to place the file in a HFS directory of your choice. (Make sure the CICS region has write permission for the directory.) The CICS-supplied sample JVM properties file, `dfjvmp.r.props`, specifies this system property as:

```
ibm.jvm.events.output=event.log
```

which creates a file called `event.log` in the directory specified by the `WORK_DIR` option.

If you use this system property to name a specific HFS file as the destination for the unresettable events information, then the information goes to that file, and is not intercepted by any class named on the `USEROUTPUTCLASS` option in the JVM profile. If the system property names the destination as `stdout` or `stderr`, then the information is intercepted by any class named on the `USEROUTPUTCLASS` option. If the system property is null, then this output is not produced at all.

**`ibm.jvm.reset.events={on}`**

Setting this system property to “on” suppresses JVM reset messages in a resettable JVM (that is, a JVM with the option **REUSE=RESET** specified in its JVM profile). These messages are normally written to the event log whenever a JVM is successfully reset.

This system property is not set in the CICS-supplied JVM properties files, so by default the function is not enabled. Note that when you include this system property in the JVM properties file with any value (“on” is one possibility), the function is enabled. To disable the function, you need to comment out or remove the system property; there is no value you can specify for the system property that disables the function.

**`ibm.jvm.resettrace.events={on}`**

Setting this system property enables the logging of reset trace events in a resettable JVM (that is, a JVM with the option **REUSE=RESET** specified in its

JVM profile). To see these events, you also need to set the `ibm.jvm.events.output` system property to enable event logging.

This system property is not set in the CICS-supplied JVM properties files, so by default the function is not enabled. Note that when you include this system property in the JVM properties file with any value (“on” is one possibility), the function is enabled. To disable the function, you need to comment out or remove the system property; there is no value you can specify for the system property that disables the function.

Reset trace events are caused by cross-heap references that are still present in out-of-scope JVM objects (garbage) in the JVM at reset time. (If the cross-heap reference is still in scope, it causes an unresettable event.) Reset trace events do not cause the JVM to be marked as unresettable and destroyed, but you should still eliminate the cross-heap references that caused them, because the trace-for-unresettable check that is required for these cross-heap references reduces the performance of the JVM. Specify the `ibm.jvm.crossheap.events` system property to log the lines of code in an application that are responsible for cross-heap references.

The activities of the Java just-in-time (JIT) compiler can interfere with the logging of reset trace events. During the development process, specify the system property `java.compiler=NONE` in the JVM properties file to turn off the JIT compiler for the JVM. (The word NONE must be in upper case.) Remember to turn the JIT compiler back on when you have finished investigating reset trace events in your application.

**`ibm.jvm.shareable.application.class.path=`**

Specifies the directory paths to be searched by the JVM for shared application classes and JAR files that are to be loaded by the shareable application class loader (SAC). When you add an application class to this class path, it is cached, and is reinitialized, rather than reloaded, if the JVM is reset. Adding application classes to this class path, rather than to the standard class path specified by the CLASSPATH option in a JVM profile, produces the best performance. The shareable application class path should be your normal choice for loading application classes in a production environment.

For worker JVMs (those with CLASSCACHE=YES in their JVM profile), the shareable application class path is taken from the JVM properties file for the master JVM that initializes the shared class cache. If it is specified in the JVM properties file for a worker JVM, it is ignored. To specify the directory paths for application classes that run in a worker JVM, use the `ibm.jvm.shareable.application.class.path` system property in the JVM properties file for the master JVM. The CICS-supplied sample JVM profile for the master JVM is DFHJVMCC, and the JVM properties file that it references is `dfjvmcc.props`.

**`ibm.jvm.trusted.middleware.class.path=`**

Specifies where the trusted middleware class loader (TMC) is to look for classes and JAR files. **Do not use this system property in a JVM properties file for CICS.** In CICS, the trusted middleware class path is built for you automatically, using the information you specify on the CICS\_DIRECTORY option in the JVM profile. Any paths you specify on the optional parameters TMPREFIX and TMSUFFIX in the JVM profile are added either at the beginning or the end of the paths constructed by CICS. If you need information about the `ibm.jvm.trusted.middleware.class.path` system property, see *Persistent Reusable Java Virtual Machine User's Guide*, SC34-6201.

**ibm.jvm.unresettable.events.level={min | max}**

Setting this system property enables the logging of unresettable events in a resettable JVM (that is, a JVM with the option **REUSE=RESET** specified in its JVM profile), and sets the level of logging required. Specifying `min` produces a list of reason codes that define the unresettable events found, and specifying `max` produces the reason codes and also a stack trace where appropriate. To see these events, you also need to set the `ibm.jvm.events.output` system property to enable event logging. The CICS-supplied JVM properties file, `dfjvjmpr.props` specifies this system property as:

```
ibm.jvm.unresettable.events.level=max
```

A frequent cause of an unresettable event is that the Java program that just ran in the JVM has performed an unresettable action. An unresettable action is when a program uses Java interfaces that modify the state of a JVM in a way that cannot be properly reset, such as changing system properties or loading a native library. The document *Persistent Reusable Java Virtual Machine User's Guide*, SC34-6201, has more information about these unresettable actions. If one or more such actions are detected during the execution of a user's Java program, the JVM is marked unresettable, and CICS destroys the JVM when the Java program has finished using it. Another possible cause of an unresettable event is if a cross-heap reference in the JVM has been found, in the course of a trace-for-unresettable check, to be still in scope (rather than in garbage). Unresettable events can also occur if there is an error in the JVM code.

The activities of the Java just-in-time (JIT) compiler can interfere with the logging of unresettable events. During the development process, specify the system property `java.compiler=NONE` in the JVM properties file to turn off the JIT compiler for the JVM. (The word `NONE` must be in upper case.) Remember to turn the JIT compiler back on when you have finished investigating unresettable events in your application.

The CICS-supplied JVM properties files also specify some special system properties that are required only for tracing in the CICS Connector for CICS TS. The CICS Connector for CICS TS allows a Java program or enterprise bean running on CICS TS to link to a CICS server program, and uses the CICS Transaction Gateway (CTG) code. These CTG trace system properties are:

```
gateway.T=off
gateway.T.entry=off
gateway.T.lines=off
gateway.T.exit=off
gateway.T.stack=off
gateway.T.trace=off
gateway.T.timing=off
```

For more information about these CTG system properties, see *Java Applications in CICS*.

---

## The sample JVM profiles and JVM properties files

*Java Applications in CICS* describes the characteristics of the CICS-supplied sample JVM profiles and JVM properties files, and the circumstances in which it would be appropriate to use each of them. For your reference, this topic provides the full text of each of the supplied sample files.

The sample files are defined with `JVMPROPS`, `LIBPATH`, `CLASSPATH`, and `WORK_DIR` parameters that use the symbols `&CICS_DIRECTORY`, `&JAVA_HOME`,

and &APPLID. As part of the CICS installation process, you will have run the DFHIJVMJ job, which is described in the *CICS Transaction Server for z/OS Installation Guide*. The DFHIJVMJ job substitutes your own values for the symbol names, and produces sample files that are tailored for your system. The text provided in this topic shows the files as they would appear after the default values had been substituted for the symbol names; that is, **cicsts31** for the &CICS\_DIRECTORY symbol, and **java142/J1.4** for the &JAVA\_HOME symbol.

When you install CICS Transaction Server for z/OS, Version 2 Release 3, the CICS-supplied sample JVM profiles are placed in the HFS directory `/usr/lpp/cicsts/cicsts31/JVMProfiles`, where `cicsts31` is the value that you chose for the CICS\_DIRECTORY symbol. The CICS-supplied sample JVM properties files are placed in the HFS directory `/usr/lpp/cicsts/cicsts31/props/`. *Java Applications in CICS* tells you what to do if you change the location of the JVM profiles or JVM properties files.

As JVM profiles and JVM properties files are HFS files, case is important. When you specify the name of a JVM profile or JVM properties file, you must enter it using the same combination of upper and lower case characters that is present in the HFS file name. The CEDA panels accept mixed case input for the JVMPROFILE field irrespective of your terminal's UCTRAN setting. However, this does not apply when values for this field are supplied on the CEDA command line, or when you are using another CICS transaction such as CEMT or CECI. If you need to enter the name of a JVM profile in mixed case when you use CEDA from the command line or when you use another CICS transaction, ensure that the terminal you use is correctly configured, with upper case translation suppressed.

The supplied sample files are as follows:

*Table 30. CICS-supplied sample JVM profiles and JVM properties files*

JVM profile	Purpose	See figure	Associated JVM properties file	See figure
DFHJVMPR	The default JVM profile if no profile is specified for a program. Resettable JVM. Does not use the shared class cache (standalone JVM).	Figure 34 on page 323	dfjjvmpr.props	Figure 35 on page 325
DFHJVMPD	Resettable JVM that uses the shared class cache (worker JVM).	Figure 36 on page 328	dfjjvmpr.props	Figure 37 on page 329
DFHJVMPD	Single-use JVM. Not recommended for enterprise beans or new applications.	Figure 38 on page 330	dfjjvmpr.props	Figure 39 on page 332
DFHJVMCC	Profile for the master JVM that initializes the shared class cache.	Figure 40 on page 333	dfjjvmcc.props	Figure 41 on page 334
DFHJVMCD (reserved for the use of CICS)	Profile for CICS-supplied system programs. Do not use for your own resource definitions. Only change as necessary.	Figure 42 on page 335	dfjjvmcd.props	Figure 43 on page 336

```

# DFHJVMPR
#
# Sample CICS JVM Profile for standalone JVM
#
# The symbol &APPLID; can be used in any of the values below
# to indicate that the applid of the CICS region should be
# substituted at run-time. This allows the use of the same profile
# for all regions, even if a different WORK_DIR (for example) is
# required, or as an alternative to the -generate option on STDOUT etc.
# With this substitution
#   STDIN=dfhjvmin.&APPLID;.data
# becomes
#   STDIN=dfhjvmin.ABCDEF.data
# for a CICS with applid ABCDEF. Applids are always upper case.
#
# ***** CICS-specific parameters *****
#
WORK_DIR=.
INVOKE_DFHJVMAT=NO
REUSE=RESET
#
# Specify the CICS and JVM install locations
#
CICS_DIRECTORY=/usr/lpp/cicsts/cicsts31/
JAVA_HOME=/usr/lpp/java142/J1.4/
#
JVMPROPS=/usr/lpp/cicsts/cicsts31/props/dfjvmp.r.props
LIBPATH=\
    /usr/lpp/cicsts/cicsts31/lib:\
    /usr/lpp/cicsts/cicsts31/ctg:\
    /usr/lpp/java142/J1.4/bin:\
    /usr/lpp/java142/J1.4/bin/classic
#
# To use the DB2 JDBC 1.2 or 2.0 drivers or the DB2 Universal
# Driver (JCC), the necessary directory containing native
# DLLs needs to be appended to LIBPATH, for example
# /usr/lpp/db2710/db2710/lib
# should be used for the JDBC 1.2 or 2.0 driver and
# /usr/lpp/db2710/db2710/jcc/lib
# for the Universal driver

```

*Figure 34. JVM options in DFHJVMPR JVM profile (Part 1 of 3)*

```

STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
# Remove comment from the line below to activate use of the
# CICS-supplied output class.
#
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#
# Uncomment and add files/directories if you wish
# to extend the automatically generated
# trusted middleware classpath.
#
# TMPREFIX=
# TMSUFFIX=
#
# For example to use the DB2 JDBC 1.2 driver, the DB2 provided zip
# file should be added to the trusted middleware classpath.
# An example is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2sqljruntime.zip
#
# An example of how to specify use of the DB2 JDBC 2.0 driver is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2j2classes.zip
#
# An example of how to specify use of the DB2 Universal Driver (JCC)
# is
# TMSUFFIX=/usr/lpp/db2710/db2710/jcc/classes/db2jcc.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_javax.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_license_cisuz.jar
#

```

*Figure 34. JVM options in DFHJVMPR JVM profile (Part 2 of 3)*

```

# ***** Java standard options *****
#
#
VERBOSE=NO
#
# Specify the path for user application classes that should
# not be cached, and should be reloaded each time the JVM is
# reused. For application classes that are to be cached, use
# use the ibm.shareable.application.class.path system property
# in the JVM property file specified by JVMPROPS=
# (unless you change this JVM profile to use the shared class
# cache, in which case use the same system property in the
# JVM properties file for the Master JVM).
#
CLASSPATH=.
#
#
# ***** Java non-standard options *****
#
Xcheck=NO
Xdebug=NO
Xms=16M
Xmx=32M
Xnoclassgc=NO
Xoss=4M
Xss=512K
Xverify=none

```

*Figure 34. JVM options in DFHJVMPR JVM profile (Part 3 of 3)*

```

#
# Properties for a standalone JVM
# -----
#
# Uncomment the following line to specify a classpath
# for Java classes that are CICS programs or Corba
# applications, but not EJB jars. If any EJB jars
# use other classes not packaged in the deployed jars
# themselves, they should be placed on this
# classpath also.
#
# ibm.jvm.shareable.application.class.path=
# /u/pathToJarOrZipFile/jarfile.jar:/u/pathToRootDirectoryForClasses
#
#
# The following lines are needed while testing applications
# for conformance with the rules for reuse of JVMs.
#
ibm.jvm.events.output=event.log
ibm.jvm.unresetable.events.level=max
#
# JNDI NameServer Configuration
# -----
#
# Note:
#   Because the necessary nameserver configuration
#   properties are likely to be common across a set
#   of CICS regions. If you wish, you can move them
#   into a file called jndi.properties and ensure
#   the directory containing this file exists in either
#   the shareable application classpath or the trusted
#   middleware classpath for all the regions wishing to
#   share the same nameserver settings.
#   By default CICS will not attempt to locate a
#   jndi.properties file. Uncomment the following line
#   to cause CICS to load jndi.properties:
#com.ibm.cics.ejs.loadjndiproperties=true
#
# EJBs must be published to a JNDI namespace so that
# the client can look them up successfully. The
# location of the JNDI nameserver where CICS will
# publish the EJBs is specified in the property:
# com.ibm.cics.ejs.nameserver

```

*Figure 35. dfjvmpr.props JVM properties file that corresponds to DFHJVMPR JVM profile (Part 1 of 3)*

```

#
# For example, if the destination system is a
# CosNaming nameserver:
# com.ibm.cics.ejs.nameserver=iiop://wibble.ibm.com:2809
#
# Some CosNaming nameservers use a port of 900.
#
# If you are using a WebSphere CosNaming JNDI service then
# you should always publish into the 'domain/legacyRoot'
# context. For example:
# com.ibm.cics.ejs.nameserver=iiop://wibble.ibm.com:2809/domain/legacyRoot
#
# Alternatively for an LDAP server:
# com.ibm.cics.ejs.nameserver=ldap://wobble.ibm.com:389
# If an LDAP nameserver is selected there are two
# additional properties to set:
#
# com.ibm.ws.naming.ldap.containerrdn
# This property *must* be set, it specifies the
# distinguished name of the System Name Space on the
# LDAP server. Your LDAP administrator will provide
# you with a suitable value for it.
#
# com.ibm.ws.naming.ldap.noderootrdn
# This property should be set if you intend to
# interoperate in an LDAP namespace with WebSphere.
# It specifies the relative distinguished
# of the legacyRoot within the System Name Space. It
# is effectively the path from the containerrdn, via the
# domainRoots tree structure down to the legacyRoot.
# Again, your LDAP system administrator can provide you
# with a suitable value.
#
# The concatenation of the containerrdn and noderootrdn
# properties determines the context where CICS will
# place a user calling `new InitialContext()`
# `legacyRoot` on the LDAP server is a suitable location
# because that is also where WebSphere/390 will be
# positioning its users that call `new InitialContext()`
#
# If noderootrdn is not specified, a call to get
# the initial context will return a context at the
# containerrdn point in the System Name Space.
# This is not a suitable location if you wish to
# interoperate on that LDAP nameserver with
# websphere. In general it is better to work with
# noderootrdn set correctly if your LDAP administrator
# has completely setup the System Name Space on your
# LDAP server.
#
# Optionally, you can have simple authentication between
# CICS and the LDAP server, this may be necessary
# depending on the access rights for the
# contexts on the LDAP server. Your LDAP administrator
# can give you suitable values for the following security
# properties:
# java.naming.security.authentication
# java.naming.security.principal
# java.naming.security.credentials
#

```

*Figure 35. dfjvmp.rprops JVM properties file that corresponds to DFHJVMPR JVM profile (Part 2 of 3)*

```

#Example LDAP configuration *with* security on:
#
#com.ibm.cics.ejs.nameserver=ldap://wobble.ibm.com:389
#com.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=cicsejbs,o=wasnaming,c=us
#com.ibm.ws.naming.ldap.noderootrdn=\
# ibm-wsnName=legacyRoot,ibm-wsnName=PLEX2,ibm-wsnName=domainRoots
#java.naming.security.authentication=simple
#java.naming.security.principal=cn=CICSAdmin
#java.naming.security.credentials=top_secret
#
# This is the set of properties you may move to a jndi.properties
# file and share amongst a group of regions.
#
# END OF JNDI NameServer Configuration
# -----
#
# CICS Connector trace properties
# -----
#
gateway.T=off
gateway.T.trace=off
gateway.T.entry=off
gateway.T.lines=off
gateway.T.exit=off
gateway.T.stack=on
gateway.T.timing=on
#
#
# JDBC Properties
# -----
#
# To avoid having to load a JDBC driver in application
# code the system property jdbc.drivers should be used to
# specify a list of named drivers separated by colons that
# the DriverManager class will attempt to load. Here is an
# example of naming the DB2 JDBC driver
# jdbc.drivers=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver
#
# DataSource naming
#
# To avoid having to hard code a dataSource path and name
# in your application the following property can be
# used. This property is used by the CICS supplied
# datasource samples.
# com.ibm.cics.datasource.path=jdbc/CICSDB2DataSource
#
#
# Enable Java 2 Security policy mechanism
# -----
#
# By default, the JVM runs without Java 2 security enabled.
# Here is an example of the properties required to enable CICS
# enterprise beans and Java applications to run with the default
# Java 2 security manager and the sample CICS security policy file:
#
#java.security.manager=default
#java.security.policy=/usr/lpp/cicsts/cicsts31/lib/security/dfjejbpl.policy
#

```

*Figure 35. dfjvmp.rprops JVM properties file that corresponds to DFHJVMPR JVM profile (Part 3 of 3)*

```

# DFHJMPC
#
# Sample CICS JVM Profile for a worker JVM
#
# The symbol &APPLID; can be used in any of the values below
# to indicate that the applid of the CICS region should be
# substituted at run-time. This allows the use of the same profile
# for all regions, even if a different WORK_DIR (for example) is
# required, or as an alternative to the -generate option on STDOUT etc.
# With this substitution
#   STDIN=dfhjvmin.&APPLID;.data
# becomes
#   STDIN=dfhjvmin.ABCDEF.data
# for a CICS with applid ABCDEF. Applids are always upper case.
#
# ***** CICS-specific parameters *****
#
WORK_DIR=.
INVOKE_DFHJVMAT=NO
JVMPROPS=/usr/lpp/cicsts/cicsts31/props/dfjjvmc.props
#
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
# Remove comment from the line below to activate use of the
# CICS-supplied output class.
#
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#
# Specify that this profile is to be used for
# Worker JVMs.
#
CLASSCACHE=YES
#
# ***** Java standard options *****
#
VERBOSE=NO
#
# Specify the path for user application classes that should
# not be cached, and should be reloaded each time the JVM is
# reused. For application classes that are to be cached, use
# the ibm.shareable.application.class.path system property
# in the JVM properties file for the Master JVM.
#
CLASSPATH=.
#
#
# ***** Java non-standard options *****
#
Xcheck=NO
Xms=16M
Xmx=32M
Xnoclassgc=NO
Xoss=4M
Xss=512K
Xverify=none

```

*Figure 36. JVM options in DFHJMPC JVM profile*

```

#
# Properties for a Worker JVM
# -----
#
# The following lines are needed while testing applications
# for conformance with the rules for reuse of JVMs.
#
ibm.jvm.events.output=event.log
ibm.jvm.unresettable.events.level=max
#
# JNDI NameServer Configuration
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
# END OF JNDI NameServer Configuration
# -----
#
# CICS Connector trace properties
# -----
#
gateway.T=off
gateway.T.trace=off
gateway.T.entry=off
gateway.T.lines=off
gateway.T.exit=off
gateway.T.stack=on
gateway.T.timing=on
#
# JDBC Properties
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
#
# Enable Java 2 Security policy mechanism
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
#

```

*Figure 37. dfjjvmpr.props JVM properties file that corresponds to DFHJVMPR JVM profile*

```

# DFHJMPS
#
# Sample CICS JVM Profile for a single-use JVM
#
# The symbol &APPLID; can be used in any of the values below
# to indicate that the applid of the CICS region should be
# substituted at run-time. This allows the use of the same profile
# for all regions, even if a different WORK_DIR (for example) is
# required, or as an alternative to the -generate option on STDOUT etc.
# With this substitution
#   STDIN=dfhjvmin.&APPLID;.data
# becomes
#   STDIN=dfhjvmin.ABCDEF.data
# for a CICS with applid ABCDEF. Applids are always upper case.
#
# ***** CICS-specific parameters *****
#
WORK_DIR=.
INVOKE_DFHJVMAT=NO
REUSE=NO
#
# Specify the CICS and JVM install locations
#
CICS_DIRECTORY=/usr/lpp/cicsts/cicsts31/
JAVA_HOME=/usr/lpp/java142/J1.4/
#
JVMPROPS=/usr/lpp/cicsts/cicsts31/props/dfjjvmops.props
LIBPATH=\
    /usr/lpp/cicsts/cicsts31/lib:\
    /usr/lpp/cicsts/cicsts31/ctg:\
    /usr/lpp/java142/J1.4/bin:\
    /usr/lpp/java142/J1.4/bin/classic
#
# To use the DB2 JDBC 1.2 or 2.0 drivers or the DB2 Universal
# Driver (JCC), the necessary directory containing native
# DLLs needs to be appended to LIBPATH, for example
# /usr/lpp/db2710/db2710/lib
# should be used for the JDBC 1.2 or 2.0 driver and
# /usr/lpp/db2710/db2710/jcc/lib
# for the Universal driver
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
# Remove comment from the line below to activate use of the
# CICS-supplied output class.
#
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#

```

Figure 38. JVM options in DFHJMPS JVM profile (Part 1 of 2)

```

# Uncomment and add files/directories if you wish
# to extend the automatically generated
# trusted middleware classpath.
#
# TMPREFIX=
# TMSUFFIX=
#
# For example to use the DB2 JDBC 1.2 driver, the DB2 provided zip
# file should be added to the trusted middleware classpath.
# An example is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2sqljruntime.zip
#
# An example of how to specify use of the DB2 JDBC 2.0 driver is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2j2classes.zip
#
# An example of how to specify use of the DB2 Universal Driver (JCC)
# is
# TMSUFFIX=/usr/lpp/db2710/db2710/jcc/classes/db2jcc.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_javax.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_license_cisuz.jar
#
# ***** Java standard options *****
#
VERBOSE=NO
#
# Specify the path for user application classes below
#
CLASSPATH=.
#
# ***** Java non-standard options *****
#
Xcheck=NO
Xdebug=NO
Xms=16M
Xmx=32M
Xnoclassgc=NO
Xoss=4M
Xss=512K
Xverify=none

```

*Figure 38. JVM options in DFHJVMPS JVM profile (Part 2 of 2)*

```

#
# Properties for a single-use JVM
# -----
#
# Uncomment the following line to specify a classpath
# for Java classes that are CICS programs or Corba
# applications, but not EJB jars. If any EJB jars
# use other classes not packaged in the deployed jars
# themselves, they should be placed on this
# classpath also.
#
# ibm.jvm.shareable.application.class.path=user.jar:user.directory
#
# JNDI NameServer Configuration
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
# END OF JNDI NameServer Configuration
# -----
#
# CICS Connector trace properties
# -----
#
gateway.T=off
gateway.T.trace=off
gateway.T.entry=off
gateway.T.lines=off
gateway.T.exit=off
gateway.T.stack=on
gateway.T.timing=on
#
# JDBC Properties
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
#
# Enable Java 2 Security policy mechanism
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
#

```

*Figure 39. dfjvmps.props JVM properties file that corresponds to DFHJVMPs JVM profile*

```

# DFHJVMCC
#
# Sample CICS JVM Profile for a master JVM
#
# The symbol &APPLID; can be used in any of the values below
# to indicate that the applid of the CICS region should be
# substituted at run-time. This allows the use of the same profile
# for all regions, even if a different WORK_DIR (for example) is
# required, or as an alternative to the -generate option on STDOUT etc.
# With this substitution
#   STDIN=dfhjvmin.&APPLID;.data
# becomes
#   STDIN=dfhjvmin.ABCDEF.data
# for a CICS with applid ABCDEF. Applids are always upper case.
#
# ***** CICS-specific parameters *****
#
WORK_DIR=.
INVOKE_DFHJVMAT=NO
REUSE=RESET
#
# Specify the CICS and JVM install locations
#
CICS_DIRECTORY=/usr/lpp/cicsts/cicsts31/
JAVA_HOME=/usr/lpp/java142/J1.4/
#
JVMPROPS=/usr/lpp/cicsts/cicsts31/props/dfjjvmcc.props
LIBPATH=\
    /usr/lpp/cicsts/cicsts31/lib:\
    /usr/lpp/cicsts/cicsts31/ctg:\
    /usr/lpp/java142/J1.4/bin:\
    /usr/lpp/java142/J1.4/bin/classic
#
# To use the DB2 JDBC 1.2 or 2.0 drivers or the DB2 Universal
# Driver (JCC), the necessary directory containing native
# DLLs needs to be appended to LIBPATH, for example
# /usr/lpp/db2710/db2710/lib
# should be used for the JDBC 1.2 or 2.0 driver and
# /usr/lpp/db2710/db2710/jcc/lib
# for the Universal driver
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
CLASSCACHE_MSGLOG=dfhjvmccmsg.log
#

```

*Figure 40. JVM options in DFHJVMCC JVM profile for the master JVM that initializes the shared class cache (Part 1 of 2)*

```

# Uncomment and add files/directories if you wish
# to extend the automatically generated
# trusted middleware classpath.
#
# TMPREFIX=
# TMSUFFIX=
#
# For example to use the DB2 JDBC 1.2 driver, the DB2 provided zip
# file should be added to the trusted middleware classpath.
# An example is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2sqljruntime.zip
#
# An example of how to specify use of the DB2 JDBC 2.0 driver is
# TMSUFFIX=/usr/lpp/db2710/db2710/classes/db2j2classes.zip
#
# An example of how to specify use of the DB2 Universal Driver (JCC)
# is
# TMSUFFIX=/usr/lpp/db2710/db2710/jcc/classes/db2jcc.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_javax.jar:\
# /usr/lpp/db2710/db2710/jcc/classes/db2jcc_license_cisuz.jar
#
# ***** Java standard options *****
#
VERBOSE=NO
#
# ***** Java non-standard options *****
#
Xcheck=NO
Xms=1M
Xmx=4M
Xnoclassgc=NO
Xoss=4M
Xss=512K
Xverify=none

```

*Figure 40. JVM options in DFHJVMCC JVM profile for the master JVM that initializes the shared class cache (Part 2 of 2)*

```

#
# Properties for a Master JVM
# -----
#
# Uncomment the following line to specify a classpath
# for Java classes that are CICS programs or Corba
# applications, but not EJB jars. If any EJB jars
# use other classes not packaged in the deployed jars
# themselves, they should be placed on this
# classpath also.
#
# ibm.jvm.shareable.application.class.path=user.jar:user.directory
#
#

```

*Figure 41. dfjjvmcc.props JVM properties file that corresponds to DFHJVMCC JVM profile*

```

# DFHJVMCD
#
# JVM Profile for use by CICS programs
#
# The symbol &APPLID; can be used in any of the values below
# to indicate that the applid of the CICS region should be
# substituted at run-time. This allows the use of the same profile
# for all regions, even if a different WORK_DIR (for example) is
# required, or as an alternative to the -generate option on STDOUT etc.
# With this substitution
#   STDIN=dfhjvmin.&APPLID;.data
# becomes
#   STDIN=dfhjvmin.ABCDEF.data
# for a CICS with applid ABCDEF. Applids are always upper case.
#
# ***** Options that may be changed *****
#
WORK_DIR=.
#
# Specify the CICS and JVM install locations
#
CICS_DIRECTORY=/usr/lpp/cicsts/cicsts31/
JAVA_HOME=/usr/lpp/java142/J1.4/
#
JVMPROPS=/usr/lpp/cicsts/cicsts31/props/dfjvmcd.props
#
# Note that the LIBPATH shown below should only be extended when
# using an output redirection class which makes use of native
# code. Do NOT remove any of the existing directories from the
# LIBPATH while carrying out this modification.
LIBPATH=\
    /usr/lpp/cicsts/cicsts31/lib:\
    /usr/lpp/cicsts/cicsts31/ctg:\
    /usr/lpp/java142/J1.4/bin:\
    /usr/lpp/java142/J1.4/bin/classic
#
# To use the DB2 JDBC 1.2 or 2.0 drivers or the DB2 Universal
# Driver (JCC), the necessary directory containing native
# DLLs needs to be appended to LIBPATH, for example
# /usr/lpp/db2710/db2710/lib
# should be used for the JDBC 1.2 or 2.0 driver and
# /usr/lpp/db2710/db2710/jcc/lib
# for the Universal driver
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
# REUSE=YES
# Setting this option to YES makes the JVM use the shared class cache:
CLASSCACHE=NO
#
# Remove comment from the line below to activate use of the
# CICS-supplied output class.
#
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
# If you specify your own output redirection class, you also need
# to specify the path to the class using the TMSUFFIX option, and
# the path to any native code using the LIBPATH option. Include
# these in this profile if CLASSCACHE=NO, or in the profile for
# the Master JVM if CLASSCACHE=YES
#
# Uncomment and add files/directories if you wish
# to extend the automatically generated
# trusted middleware classpath.
#
# TMPREFIX=
# TMSUFFIX=
#

```

```

# ***** Java non-standard options *****
#
Xms=16M
Xmx=32M
Xoss=4M
Xss=512K
# *****
# ***** Do not change any of the options shown below *****
# *****
INVOKE_DFHJVMAT=NO
#
# ***** Java standard options *****
#
VERBOSE=NO
#
# ***** Java non-standard options *****
#
Xcheck=NO
Xdebug=NO
Xnoclassgc=NO
Xverify=none

```

Figure 42. JVM options in DFHJVMCD JVM profile (Part 2 of 2)

```

#
# Properties for a JVM used by CICS programs
# -----
#
# Uncomment the following line to specify a classpath
# for Java classes that are CICS programs or Corba
# applications, but not EJB jars. If any EJB jars
# use other classes not packaged in the deployed jars
# themselves, they should be placed on this
# classpath also.
#
# ibm.jvm.shareable.application.class.path=user.jar:user.directory
#
# The following lines are needed while testing applications
# for conformance with the rules for reuse of JVMs.
#
ibm.jvm.events.output=event.log
ibm.jvm.unresettable.events.level=max
#
# JNDI NameServer Configuration
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
# END OF JNDI NameServer Configuration
# -----
#
# Enable Java 2 Security policy mechanism
# -----
# [as for the supplied sample JVM properties file DFHJVMPR]
# ...
#

```

Figure 43. dfjjvmcd.props JVM properties file that corresponds to DFHJVMCD JVM profile

---

## Chapter 21. CICS startup

This chapter describes how to start up CICS in the CICS region. Depending on your system environment, you can start the CICS job from a procedure by using the START command, or you can submit the CICS startup job stream through the internal reader. This chapter gives an example of each of these methods. For an example of a batch job that you can submit through the internal reader, see “A sample CICS startup job” on page 340. “A sample CICS startup procedure” on page 360 gives an example of a cataloged procedure suitable for starting CICS as a started task.

When you run the startup job, you start a process called **CICS system initialization**. This process must finish before you run any transactions. Completion of CICS initialization is shown by the following message at the system console:

```
DFHSI1517 - applid: Control is being given to CICS.
```

CICS initialization involves many activities, some of which are:

- Obtaining the required storage for CICS execution from the private area in the CICS address space, above and below the 16MB line.
- Setting up CICS system parameters for the run, as specified by the system initialization parameters.
- Loading and initializing the CICS domains according to the start option specified by the START= system initialization parameter.
- Loading the CICS nucleus with the required CICS modules.
- Installing CICS resource definitions by:
  - Loading, from the CSD, the groups of resources specified by the GRPLIST= system initialization parameter
  - Loading the control tables specified by system initialization parameters
- If XRF=YES is specified, signing on to the CICS availability manager (CAVM) to check that it is possible to continue initialization and perform the role requested, that is, as an active or alternate CICS region.
- Opening the data sets necessary for initialization, including any needed for backout if the previous run of your CICS region was not shut down normally. Data sets may also be opened for backout even after a successful shutdown, if they had suffered backout failures before the CICS shutdown, because failed backouts are retried at emergency restart.
- Opening BSAM sequential devices as required in the terminal control table (TCT).

Also, if you are operating CICS with CICS recovery options, backout procedures may be used to restore recoverable resources to a logically consistent state. Briefly, backout occurs if you start CICS in one of the following ways:

- With START=AUTO and CICS detects that the previous shutdown was immediate or uncontrolled. With SDTRAN, an immediate shutdown does not always leave in-flight units of work to be backed out. Also, even if there were no in-flight UOWs, it is possible (although rare) that there were backout-failed UOWs for which backout will be retried.
- With START=STANDBY and XRF=YES, and a takeover occurs.

For background information about backout, and recovery and restart, see the *CICS Recovery and Restart Guide*.

In the final stages of initialization, a set of programs can be executed as specified in a program list table (PLT). You specify the suffix of the PLT you want by means of the PLTPI parameter in the SIT. Initialization PLT programs run under control of a CICS task in CICS key. Their storage is in CICS-key storage, and protected from overwriting by other transactions.

# The PLTPI resource managers start up in the following sequence when specified in  
# the system initialization table:

1. CPSMCONN=CMAS
2. CPSMCONN=LMAS
3. CPSMCONN=WUI
4. DBCTLCON=YES
5. DB2CON=YES
6. MQCON=YES

# This sequence only applies to the above system initialization parameters and not for  
# PLTPI programs in general.

For more information about storage protection, see “Storage protection” on page 354. For programming information about writing PLT programs, see the *CICS Customization Guide*.

If you are running CICS with DB2, you can specify the DB2 subsystem ID to be used at PLT startup by the INITPARM system initialization parameter, as follows:

```
INITPARM=(DFHD2INI='yyyy')
```

where yyyy is the 4-character DB2 subsystem ID. The value must conform to MVS JCL rules about special characters. You cannot use the ID of a data sharing group of DB2 subsystems on the INITPARM system initialization parameter — you must specify the ID of a single DB2 subsystem. If you want to use the INITPARM system initialization parameter to specify a DB2 subsystem, leave blank both the DB2GROUPLD and the DB2ID in the installed DB2CONN definition. An ID specified in either of these attributes of the DB2CONN definition overrides an ID specified on the INITPARM system initialization parameter.

---

## Using the sample startup job stream

You can use the sample job control statements shown in Figure 44 on the following pages to initialize a CICS region. The sample job stream shown in Figure 44 specifies START=AUTO and XRF=YES to start an active CICS region. The notes that follow Figure 44 explain which statements are unique to XRF, and can therefore be omitted if you want to run with XRF=NO.

The sample startup job stream is based on the system initialization parameters contained in the CICS-supplied sample table, DFHSIT6\$.

For more information about the DD statements in this job stream that are needed by CICS and IMS, see the appropriate chapter in Part 2, “Defining data sets,” on page 21.

JCL similar to that in Figure 44 on page 340 is supplied as a sample startup procedure, DFHSTART, in the CICSTS31.CICS.SDFHINST library. You can use the DFHSTART procedure to start the CICS regions, or as a basis for your own startup procedures. For information about the DFHSTART procedure, see the *CICS*



## A sample CICS startup job

```
/*
/** 1 The JOB statement
//CICSRUN JOB accounting info,name,CLASS=A,
// MSGCLASS=A,MSGLEVEL=(1,1)
/**
/** 2 The JOBPARM statement
/*JOBPARM SYSAFF=sysid
/*
//*****
//***** EXECUTE CICS *****
//*****
/**
/** 3 The EXEC CICS=DFHSIP statement
//CICS EXEC PGM=DFHSIP,REGION=240M,
/** 4 SIT parameters specified on PARM parameter
// PARM=('SIT=6$',
// 'DSALIM=6M,EDSALIM=120M',
// 'RENTPGM=PROTECT,STGPROT=YES',
// 'START=AUTO,SI')
/**
/** 5 SIT parameters specified on the SYSIN data set
//SYSIN DD *
GRPLIST=(DFHLIST,userlist1,userlist2),
LPA=YES,
APPLID=CICSHTH1,
DFLTUSER=CICSUSER, The default userid
MXT=30, Maximum number of user tasks is 30
INITPARM=(DFHDBCON='01',DFHD2INI=('MYDB')),
Pass DFSPZP01 suffix to DBCTL connect program
Connect to DB2 subsystem MYDB
ISC=YES, Include intersystem communication program
IRCSTRT=YES, Start interregion communication
.END
/*
/** 6 The STEPLIB library
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR
// DD DSN=CICSTS31.CICS.SDFJAUTH,DISP=SHR
// DD DSN=CEE.SCEERUN2,DISP=SHR
// DD DSN=CEE.SCEERUN,DISP=SHR
/**
/** 7 The CICS library (DFHRPL) concatenation
//DFHRPL DD DSN=CICSTS31.CICS.SDFHLOAD,DISP=SHR
/** Your application library
// DD DSN=your.prog.library,DISP=SHR
/** Your CICS control tables library
// DD DSN=your.table.library,DISP=SHR
/** The Language Environment runtime data sets
// DD DSN=CEE.SCEECICS,DISP=SHR
// DD DSN=CEE.SCEERUN2,DISP=SHR
// DD DSN=CEE.SCEERUN,DISP=SHR
/** The Debug Tool runtime library
// DD DSN=EQA.SEQAMOD,DISP=SHR
/** 7a The DB2 load library
// DD DSN=SYS2.DB2.SDSNLOAD,DISP=SHR
/** 8 Auxiliary temporary storage data sets
//DFHTEMP DD DSN=CICSTS31.CICS.CNTL.CICSHTH1.DFHTEMP,DISP=SHR
/**
/** 9 Intrapartition data sets
//DFHINTRA DD DSN=CICSTS31.CICS.CNTL.CICSHTH1.DFHINTRA,DISP=SHR
/**
```

Figure 44. CICS startup job stream 1/3

#

```
/** 10 The auxiliary trace data sets
//DFHAUXT DD DSN=CICSTS31.CICS.CICSHTH1.DFHAUXT,DISP=SHR
//DFHBUXT DD DSN=CICSTS31.CICS.CICSHTH1.DFHBUXT,DISP=SHR
/**
/** 11 Extrapartition data sets
//DFHCXRF DD SYSOUT=*
//LOGUSR DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//MSGUSR DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//PLIMSG DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)
//COUT DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)
//CEEMSG DD SYSOUT=A
//CEEOUT DD SYSOUT=A
/**
/** 12 The CICS local catalog data set
//DFHLCD DD DSN=CICSTS31.CICS.CICSHTH1.DFHLCD,DISP=SHR
/** 13 The CICS global catalog data set
//DFHGCD DD DSN=CICSTS31.CICS.CICSHTH1.DFHGCD,DISP=SHR,
// AMP=('BUFND=33,BUFNI=32,BUFSP=319488')
/**
/** 14 The CAVM data sets - XRF
//DFHXRMMSG DD DSN=CICSTS31.CICS.CNTL.CICSHTH1.DFHXRMSG,DISP=SHR
//DFHXRCTL DD DSN=CICSTS31.CICS.CNTL.CICSHTH1.DFHXRCTL,DISP=SHR
/**
/** 15 The transient data destination data set (CXRF)
//DFHCXRF DD SYSOUT=A
/**
/** 16 The CICS transaction dump data sets
//DFHDMPA DD DSN=CICSTS31.CICS.CICSHTH1.DFHDMPA,DISP=SHR
//DFHDMPB DD DSN=CICSTS31.CICS.CICSHTH1.DFHDMPB,DISP=SHR
/**
/** 17 MVS system dump data sets
//SYSABEND DD SYSOUT=*
//SYSMDUMP DD DSN=SYS1.SYSMDP00,VOL=SER=valid,SPACE=(CYL,(1,1)),
// DISP=OLD,UNIT=3380
/** SYSUDUMP DD SYSOUT=*
/**
/** 17a Default stdout file for C-language system services used by CICS
//SYSPRINT DD SYSOUT=*
/** 18 The CICS system definition data set
//DFHCSD DD DSN=CICSTS31.CICS.DFHCSD,DISP=SHR
/**
/** 19 The CICS BTS local request queue data set
//DFHLRQ DD DSN=CICSTS31.CICS.DFHLRQ,DISP=SHR
/**
/** 20 The EJB directory and object store data sets
//DFHEJDIR DD DSN=CICSTS31.CICS.DFHEJDIR,DISP=SHR
//DFHEJOS DD DSN=CICSTS31.CICS.DFHEJOS,DISP=SHR
```

Figure 45. CICS startup job stream 2/3

```

/* 21 The CMAC data file
//DFHMACD DD DSN=CICSTS31.CICS.DFHMACD,DISP=SHR
/*
/* 22 The CDBM group command file
//DFHDBFK DD DSN=CICSTS31.CICS.DFHDBFK,DISP=SHR
/*
/* 23 FILEA & other permanently allocated data sets
/* (The FILEA DD statement below overrides the CSD definition in
/* group DFHMROFD)
//FILEA DD DISP=SHR,
// DSN=CICSTS31.CICS.CICSHTH1.FILEA
/*
//APPLICA DD DSN=CICSTS31.CICS.CICSHTH1.APPLICA,DISP=SHR
//APPLICB DD DSN=CICSTS31.CICS.CICSHTH1.APPLICB,DISP=SHR
/*
/*
/* 24
//PRINTER DD SYSOUT=A,DCB=BLKSIZE=125,OUTLIM=0
//CARDIN DD *
:
:user transactions input from sequential terminal \
:
\CESF GOODNIGHT\
/*

```

Figure 46. CICS startup job stream 3/3

## Notes:

### 1 The JOB statement

The JOB statement specifies the accounting information that you want to use for this run of CICS. For example:

```

//CICSRUN JOB 24116475,userid,MSGCLASS=A,MSGLEVEL=(1,1),
// CLASS=A,NOTIFY=userid

```

### 2 The JOBPARM statement

The JES JOBPARM statement is included to identify the MVS image on which this CICS (the active CICS region) is to run. If the alternate CICS region is to run on a different MVS image, the job stream for the alternate CICS region specifies the system identifier of the other MVS image.

### 3 The EXEC PGM=DFHSIP statement

DFHSIP is the program that starts CICS initialization.

CICS does not support more than one EXEC PGM=DFHSIP job step in the same MVS job.

The EXEC statement contains the REGION parameter to define the size of CICS MVS region. In this example, the value is set to 240, requesting MVS to allocate to the job all 16MB of private storage below the 16MB line, and an extended region size of 240MB.

You determine how much of the allocated private storage you want for the CICS dynamic storage areas, and how much CICS is to leave for demands on operating system storage, by setting values for the DSALIM and EDSALIM system initialization parameters. After obtaining the amount of space required for the DSAs

from the total defined by the REGION parameter, the remaining storage is available to meet demands for operating system storage.

In our sample job stream, these system initialization parameters are specified in the PARM parameter (see the next topic).

For more details about the REGION parameter and CICS storage, see “Storage requirements for a CICS region” on page 352.

If you are running CICS with RACF support, see the *CICS RACF Security Guide* for information about RACF-related parameters.

#### **4 SIT options on the PARM parameter**

You can use the PARM parameter of the EXEC statement to specify system initialization parameters as shown.

The information passed by the PARM parameter is limited to 100 characters. This limit includes all commas, but excludes the apostrophes delimiting the PARM strings, and excludes the opening and closing parentheses delimiting the PARM parameter. (Internal parentheses enclosing system initialization operands **are** included.) If 100 characters are not sufficient for the system initialization parameters you want to provide at startup, indicate continuation by ending the PARM field with the “SYSIN” or “CONSOLE” control keywords (or “SI” or “CN” for short). If you specify SYSIN, system initialization parameters are read from the SYSIN data set; if you specify CONSOLE, CICS prompts you to enter parameters through the console. However, if all of your runtime system initialization parameters are in the PARM parameter, you can end the PARM field simply without any control keywords, or by the .END control keyword.

In our example, DFHSIT6\$ is the SIT selected, and CICS system initialization uses the values in that table, modified by the system initialization parameters supplied in the PARM field and the SYSIN data set. For this example, the following system initialization parameters are provided in the PARM parameter:

##### **RENTPGM**

Storage for the read-only DSAs, RDSA and ERDSA, is obtained from key-0, non-fetch protected storage by using the default PROTECT option on the RENTPGM system initialization parameter. You are recommended to specify RENTPGM=NOPROTECT for development CICS regions and RENTPGM=PROTECT for production CICS regions. For more information about the RENTPGM parameter, see page “RENTPGM” on page 227.

##### **STGPROT**

Storage protection is obtained for this run of CICS by specifying YES on the STGPROT system initialization parameter. Before using this parameter, check with the *CICS Transaction Server for z/OS Program Directory* that you have the required hardware and software. See page “STGPROT” on page 242 for details about the STGPROT parameter itself.

##### **START**

START=AUTO is normally the type of start you would select for a production CICS system, allowing CICS to determine the class of start to be performed (warm, emergency, or cold).

If you are running CICS with XRF, START=AUTO causes CICS to initialize as an active CICS region. To request initialization of CICS as an alternate CICS region, specify XRF=YES and START=STANDBY.

For information about the types of CICS startup and shutdown in an XRF environment, see the *CICS Operations and Utilities Guide*.

- SI** The PARM statement is terminated with SI (short for SYSIN), to tell CICS to continue reading overrides from the SYSIN data set.

## **5** SYSIN data stream

You can include the SYSIN data set inline as part of the job stream. System initialization parameters entered in the SYSIN data set replace any, for the same keyword, that were entered in the PARM parameter. If you include the same parameter more than once, the **last** value read is the value used for initialization except for INITPARM. If you specify the INITPARM keyword and its parameters more than once, each one is accepted by CICS, for example:

```
* The following INITPARM parameters are for DBCTL and a user program
INITPARM=(DFHDBCON='XX,DBCON2',userprog='a,b,c')
* The following INITPARM parameter is for DB2
INITPARM=(DFHD2INI='DBA2')
```

Unless you explicitly code the system initialization control keyword **CONSOLE**, CICS stops reading system initialization parameters when it reaches the end of SYSIN or a **.END** control keyword.

In the sample job, **CONSOLE** is not coded in either PARM or SYSIN. The **.END** control keyword is the last entry in SYSIN, so CICS does not prompt through the console for further system initialization parameters. After reading the SYSIN data set, CICS loads the specified SIT, applies any system initialization parameters supplied in the PARM field and the SYSIN data set, and begins the initialization process.

The SYSIN data set in our example includes several system initialization parameters, as follows:

### **GRPLIST**

The group list defined in DFHSIT6\$ is DFHLIST, the IBM-defined list that is generated when you initialize the CSD using the DFHCSDUP INITIALIZE command. DFHLIST contains only the standard IBM-defined resource definitions required by CICS. One of the group lists specified on the GRPLIST system initialization parameter must contain those resource definitions required by CICS. You can do this either by including the resource definitions in one of your own group lists that you specify, or by specifying the DFHLIST explicitly, as shown. Your own group lists (userlist1 and userlist2 as shown) should contain all the resource definitions generated by your installation for your applications that are required for this CICS run. In addition, your group lists should contain any definitions required for IBM program products that you are using, such as COBOL or DB2.

- LPA** The SIT specifies that modules are not to be used from the link pack area; LPA=YES in SYSIN specifies that modules are to be used from the LPA in this run.

### **APPLID**

The applid for this CICS region is CICSHTH1. If you want this CICS region to use VTAM for terminal access or ISC communication, this applid must be defined to VTAM.

If you want this CICS region to use XRF, you must define both a generic and specific applids; for example, by:

APPLID=(CICSID,CICSHTH1),

CICSID is the generic applid of this CICS region, and CICSHTH1 is the specific applid of the active CICS region. With XRF=YES, the active and alternate CICS regions share the same generic applid, but have different specific applids.

The specific applid can be useful for naming those data sets that are unique (for example, dump data sets). Where necessary, it can be used as the second-level qualifier to distinguish the data sets of the active and alternate CICS regions.

### **CICSSVC**

245 is the CICS type 3 SVC number installed in the LPA, and defined to MVS in an SVC Parm statement. For more information about the CICSSVC parameter, see page “CICSSVC” on page 172.

For guidance information about installing the CICS SVC in the LPA, and defining it to MVS, see the *CICS Transaction Server for z/OS Installation Guide*.

### **DFLTUSER**

CICSUSER is the default userid specified to RACF. During startup, CICS tries to sign on the default userid. If it cannot be signed on (for example, if not defined), CICS issues a message and terminates CICS initialization. After the valid default userid is signed on, its security attributes are used for all CICS terminal users who do not sign on with the CESN transaction. If the default userid is defined to RACF with a CICS segment, the operator attributes in that segment are also used for users who do not sign on.

**MXT** The maximum number of user tasks is limited to 30 for this run. For information about what tasks are included in the MXT parameter, see page “MXT” on page 215.

### **INITPARM**

Passes parameters to programs. In this example the DBCTL suffix (01) of DFSPZPxx is passed to DFHDBCON, and CICS is told to connect to the DB2 subsystem MYDB. (You cannot use the ID of a data sharing group of DB2 subsystems on the INITPARM system initialization parameter — you must specify the ID of a single DB2 subsystem.) Note that if you want to use the INITPARM system initialization parameter to specify a DB2 subsystem, leave blank both the DB2GROUPID and the DB2ID in the installed DB2CONN definition. An ID specified in either of these attributes of the DB2CONN definition overrides an ID specified on the INITPARM system initialization parameter.

**ISC** Include the intersystem communication program, DFHISP, in order to use interregion communication (IRC).

### **IRCSTRT**

This CICS is running with MRO, and interregion communication is started during initialization.

## **6 STEPLIB library**

STEPLIB is the DDNAME of the library containing the modules loaded by the operating system. DFHSIP, which is loaded from STEPLIB, must receive control in an authorized state, so each partitioned data set (library) concatenated in STEPLIB must be individually APF-authorized. In this sample job stream, the CICS authorized library is CICSTS31.CICS.SDFHAUTH. CICSTS31.CICS.SDFJAUTH is also

included, which is required for Java support, and must also be APF-authorized. If you do not require Java support, you need not include this library.

Placing user-written modules into an APF-authorized library may violate your security and integrity rules. The CICSTS31.CICS.SDFHAUTH library (and the CICSTS23.CICS.SDFJAUTH library, if used) should not be included in the MVS link list, so that you can protect the libraries and thereby ensure that only approved users are able to execute the DFHSIP module. When you define your STEPLIB DD statement, remember that all other libraries concatenated with the CICSTS31.CICS.SDFHAUTH library or the CICSTS23.CICS.SDFJAUTH library must also be APF-authorized (for example, IMS.RESLIB). This is because, if any of the libraries in a STEPLIB concatenation are not authorized, MVS regards all of them as unauthorized. (If there is a non-authorized library in the STEPLIB concatenation, CICS fails to start up, and issues message DFHKE0101 to indicate that the DFHSIP module is not in an APF-authorized library.) For information about authorizing access to CICS data sets, see the *CICS RACF Security Guide*.

The pregenerated DFHSIP module, which has been link-edited with the authorized attribute (SETCODE AC(1)), is supplied in CICSTS31.CICS.SDFHAUTH.

# You also need the Language Environment runtime libraries, CEE.SCEERUN and  
# CEE.SCEERUN2, in the STEPLIB concatenation (or the MVS linklist) to run  
# COBOL, PL/I, C and C++, and Java programs that run in a JVM, under Language  
# Environment. The order of the SCEERUN and SCEERUN2 libraries in relation to  
# each other is important. SCEERUN2 must appear in the concatenation before  
# SCEERUN. Like SDFHAUTH, SCEERUN and SCEERUN2 must be an  
# APF-authorized library.

The CICS DB2 attachment facility has to load the DB2 program request handler, DSNAPRH. To do this, the DB2 library DSNxxx.SDSNLOAD library should be placed in the MVS linklist, or added to the STEPLIB concatenation of the CICS job.

## **7 CICS module load library (DFHRPL) concatenation**

DFHRPL is the DD name of the library that contains modules loaded by CICS. Protect individually the partitioned data sets constituting this library to prevent unapproved or accidental modification of their contents. The DFHRPL concatenation must include the library containing your CICS application programs, shown in our example as “your.prog.library”, and your CICS control tables, shown in our example as “your.table.library”.

In this sample job stream, the CICS-supplied library is CICSTS31.CICS.SDFHLOAD, which must be included in the CICS DFHRPL library concatenation. The CICSTS31.CICS.SDFHLOAD library contains only programs that run in problem state and should not be authorized.

### **Language Environment runtime library requirements**

You are recommended to use the services of Language Environment, which provides a common runtime environment for IBM implementations of assembler and those high-level languages (HLLs) supported by CICS, namely COBOL, PL/I, C, and C++.

# To use the Language Environment runtime libraries to support all your program  
# languages, add the Language Environment runtime libraries SCEERUN2 and  
# SCEERUN to the DFHRPL DD concatenation. The SCEERUN2 library must appear

# in the concatenation before SCEERUN. If you are running COBOL programs, also  
# add Language Environment runtime library SKEECICS to DFHRPL. The  
# SKEECICS library must be concatenated before the SCEERUN library. Remove  
# any libraries that contain runtime routines from earlier versions of COBOL, PL/I, and  
# C/C++ from the DFHRPL DD concatenation.

### **Debug Tool library**

If you are using Debug Tool to debug CICS applications, include the Debug Tool library SEQAMOD in DFHRPL. For information about using Debug Tool with CICS, see the *CICS Application Programming Guide*.

### **7a DB2 requirements**

Generally, you do not need to include any DB2 libraries in the DFHRPL DD statement. If you do need DB2 libraries in the DFHRPL concatenation for an application, they should be placed after the CICS libraries. For example, you need SDSNLOAD in the DFHRPL to support those applications that issue dynamic calls to the DB2 message handling module, DSNTIAR, or the later DSNTIA1, both of which are shipped in SDSNLOAD. DSNTIA1 is loaded by applications programs that include the DB2 application stub DSNTIAC, which issues an EXEC CICS LOAD command for program DSNTIAC.

### **8 Auxiliary temporary storage (DFHTEMP) data sets**

Define this data set if you want to save data to use later. The temporary storage queues used, identified by symbolic names, exist until explicitly deleted. Even after the originating task is deleted, temporary data can be accessed by other tasks, through references to the symbolic name under which it is stored.

For details of how to define these data sets, and for information about space calculations, see Chapter 4, “Setting up the temporary storage data set,” on page 35.

If you are using temporary storage data sharing, you should ensure that you start the temporary storage server before it is required by the CICS regions.

For more information about the temporary storage server and temporary storage data sharing, see Chapter 24, “Setting up and running a temporary storage server,” on page 369

### **9 Intrapartition transient data (DFHINTRA) data sets**

The transient data intrapartition data set is used for queuing messages and data within the CICS region.

For information about how to define these data sets, and about space calculations, see “Defining intrapartition data sets” on page 42.

### **10 Auxiliary trace (DFHAUXT and DFHBUXT) data sets**

Define one or both of these sequential data sets, if you want to use auxiliary trace. If you define automatic switching for your auxiliary trace data sets, define both data sets. If you define only one data set, its DD name must be DFHAUXT.

For details of how to define these data sets, see Chapter 9, “Setting up and using auxiliary trace data sets,” on page 95.

The auxiliary trace data sets in this job stream are unique to the active CICS region, and as such are identified in our example by using the specific applid of the active CICS region (CICSHTH1) as a second-level qualifier. If you are using auxiliary trace with XRF=YES, the alternate CICS region also needs its own trace data sets. These could be identified by the specific applid of the alternate CICS region, for example, CICSHTH2.

If you allocate and catalog the auxiliary trace data sets on disk as shown in Figure 16 on page 96, you can define them to CICS in the startup job stream using the following DD statements:

```
//DFHAUXT DD DSN=CICSTS31.CICS.applid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
//DFHBUXT DD DSN=CICSTS31.CICS.applid.DFHBUXT,DCB=BUFNO=n,DISP=SHR
```

If you specify BUFNO greater than 1, you can reduce the I/O overhead involved in writing auxiliary trace records. A value between 4 and 10 can greatly reduce the I/O overhead when running with auxiliary trace on.

## **11** Extrapartition transient data queues

LOGA, CSSL, and CPLI are examples of extrapartition transient data queues.

- LOGA defines a user data set used by the CICS sample programs.
- CSSL defines the data set used by a number of CICS services.
- CPLI is used only when you are running PL/I application programs. Here, CPLII defines the data set to which both statistics and messages, and PL/I dumps are directed.
- CCSO is used as an output queue only when you are running C/370™ application programs.
- CESO is used as an error queue only when you are running application programs under Language Environment.

Sample definitions of the queues used by CICS are supplied in group DFHDCTG. DFHDCTG is unlocked, so you can alter the definitions before installation.

## **12** The CICS local catalog data set

The CICS local catalog is used by the CICS domains to save some of their information between CICS runs, and to preserve this information across a cold start. The local catalog is not shared by any other CICS system. If you are running CICS with XRF, define a unique local catalog for the active CICS region, and another for the alternate CICS region. For details of how to create and initialize a CICS local catalog, see Chapter 8, “Setting up and using catalog data sets,” on page 85.

## **13** The CICS global catalog data set

There is only one global catalog, which is passively shared by the active and alternate CICS regions.

For details of how to create and initialize a CICS global catalog, see Chapter 8, “Setting up and using catalog data sets,” on page 85.

This sample job illustrates the use of the AMP parameter on the DD statement. Specifying this parameter, with its buffer subparameters, can help to improve restart

and shutdown time. This example is based on the recommended DEFINE CLUSTER statements shown in Figure 13 and the associated notes given under **4** on page “Defining the global catalog” on page 86. The values given are the minimum values suitable for these parameters and should not be reduced.

#### **14 The CAVM data sets**

These data sets are required when you are running CICS with XRF. They are actively shared by the active and the alternate CICS regions. For details of how to create and initialize the CICS availability data sets, see Chapter 11, “Defining the CICS availability manager data sets,” on page 105.

#### **15 The DFHCXRF transient data set (CXRF)**

This transient data destination is used by CICS as the target for messages sent to any transient data destination before CICS has completed intrapartition transient data initialization. It is particularly necessary in an XRF environment for use in an alternate CICS region before takeover has occurred, during the period when transient data initialization is suspended. For more information about the DFHCXRF data set, see “The DFHCXRF data set” on page 46.

#### **16 CICS transaction dump data sets**

CICS records transaction dumps on a sequential data set, or pair of sequential data sets, tape or disk. The data sets must be defined with the DD names DFHDMPA and DFHDMPB, but if you define only one data set, its DD name must be DFHDMPA. CICS always attempts to open at least one transaction dump data set during initialization.

For details about how to define CICS transaction dump data sets and how they are used, see Chapter 10, “Defining dump data sets,” on page 99.

The transaction dump data sets in this job stream are unique to the active CICS region, and as such are identified by using the specific applid of the active CICS region (DBDCCIC1) as a second-level qualifier. The alternate CICS region needs its own transaction dump data sets, and these could be identified by using the specific applid of the alternate CICS region (DBDCCIC2) as a second-level qualifier.

#### **17 MVS system dump data sets**

Use a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement to direct MVS to produce a dump.

In the sample job stream (Figure 44 on page 340), the SYSABEND DD statement directs a formatted dump to a printer, and the SYSMDUMP DD statement saves an unformatted dump to the SYS1.SYSMDP00 data set on disk.

MVS produces the requested dump if either of the following is true:

- CICS terminates abnormally.
- CICS starts to terminate abnormally, but system recovery procedures enable CICS to terminate normally.

The dump DD statements for requesting dumps are:

##### **SYSABEND DD statement**

Produces a dump of user and system areas; this dump contains all the

areas dumped in a SYSUDUMP plus the local system queue area (LSQA), including subpools 229 and 230, and the input/output system (IOS) control blocks for the failing task. The dump is formatted, so that it can be printed directly.

#### **SYSMDUMP DD statement**

Produces a dump of the system areas and the program's address space. The dump is unformatted and machine-readable; to be used, it must be printed by the interactive problem control system (IPCS).

**Note:** The SYSMDUMP DD statement must specify a magnetic tape unit or a direct access device.

To write more than one SYSMDUMP dump in the same data set on tape, specify the following:

- DSNAME=SYS1.SYSMDPxx where xx is 00 through FF. SYSMDPxx is a preallocated data set that you must initialize with an end-of-file (EOF) mark on the first record.
- DISP=SHR.

You can ask MVS to write additional dumps only if you off-load any previous dump and write an EOF mark at the beginning of the SYS1.SYSMDPxx data set. To accomplish this, your MVS installation must install an exit routine for message IEA993. For information on this installation exit routine, see the *OS/390 MVS Installation Exits* manual.

#### **SYSUDUMP DD statement**

Produces a dump of user areas. The dump is formatted, so that it can be printed directly.

The dump contents are as described only when you use the IBM-supplied defaults for the dumps. The contents of these dumps can be set during MVS system initialization and can be changed for an individual dump in the ABEND macro instruction, in a CHNGDUMP command, and by a SLIP command. For details, see the *OS/390 MVS Initialization and Tuning Guide* manual.

Dumps are optional; use a dump DD statement only when you want to produce a dump.

For information about how defining these MVS system dump data sets, and about printing dumps from them, see the *OS/390 MVS JCL Reference* manual. For information about how to interpret dumps, see the *OS/390 MVS Diagnosis: Tools and Service Aids* manual.

#### **17a Default stdout file for C-language system services used by CICS**

The SYSPRINT statement is opened as the stdout file by C-language system service routines that are used by CICS (such as system SSL.) If it is omitted, multiple sysout files might be dynamically allocated by the operating system instead.

#### **18 The CICS system definition file**

The system definition file (CSD) is required by CICS to hold some resource definitions.

You may want to provide job control DD statements for the CSD. If you do, the CSD data set is allocated at the time of CICS job step initiation, and **remains allocated for the duration of the CICS job step**.

On the other hand, you may prefer to use dynamic allocation of the CSD. For dynamic allocation, do **not** specify a DD statement for the CSD. Specify the data set name (DSNAME) and the data set disposition (DISP) either in a SET FILE command or in the SIT (as parameters CSDDSN and CSDDISP). CICS uses the DSNAME and DISP to allocate the file as part of OPEN processing.

For information about creating and initializing the CSD, see Chapter 7, “Setting up the CICS system definition data set,” on page 63.

If you are running CICS with XRF, particularly in a multi-MVS environment, there are special considerations concerning CSD sharing; these considerations are discussed under “Sharing the CSD in non-RLS mode” on page 68.

### **19 The CICS BTS local request queue data set**

DFHLRQ is a file-control-managed VSAM key-sequenced data set (KSDS), used by CICS business transaction services (BTS). The IBM-supplied file resource definition for DFHLRQ is supplied in CSD group DFHCBTS, which is automatically included in DFHLIST group list when you initialize or upgrade your CSD. The file definition specifies that it is to be opened on first reference, which occurs at the end of CICS initialization when it is opened by BTS. CICS issues warning messages DFHFC0951 and DFHSH0109 if the data set is not found. Although CICS continues running without this data set, you are recommended to define the DFHLRQ data set, even if you are not using BTS. For information about DFHLRQ, see the *CICS Business Transaction Services* manual.

### **20 The EJB directory (DFHEJDIR) and the EJB object store (DFHEJOS)**

DFHEJDIR is a VSAM key-sequenced data set (KSDS) that contains a request streams directory, which must be shared by all the regions (listeners and AORs) in a logical EJB server. Request streams are used in the distributed routing of method requests for enterprise beans and CORBA stateless objects.

DFHEJOS is a VSAM key-sequenced data set (KSDS) that contains details of stateful session beans that have been passivated. It must be shared by all the AORs in the logical EJB server.

For information about defining the EJB directory and object store, see Figure 27 on page 139 and Figure 26 on page 138.

### **21 CMAC data file (DFHCMACD)**

DFHCMACD is a VSAM key-sequenced data set (KSDS) that is used by the CMAC transaction to provide online descriptions of CICS messages and codes. Before its first use it must be defined and loaded as a KSDS data set.

Chapter 14, “Defining the CMAC messages data set,” on page 135 describes DFHCMACD in greater detail.

### **22 CDBM group command file (DFHCDBK)**

DFHDBFK is a VSAM key-sequenced data set (KSDS) that is used by the CDBM transaction to store Group commands. Before its first use it must be defined as a KSDS data set. The DFHDBFK DD statement is only required if you intend to use the command storage functions of the CDBM transaction.

“Defining the CDBM GROUP command data set” on page 131 describes DFHDBFK in greater detail.

### **23 Sample program file (FILEA) and other permanently allocated data sets**

You may want to provide job control DD statements for those user files that are defined in the CSD (if you are using RDO) or in a file control table (for BDAM files only). If you do, the data sets are allocated at the time of CICS job step initiation, and **remain allocated for the duration of the CICS job step**. FILEA, the distributed library containing the data for the sample application programs, is included in our startup job stream as an example of direct allocation by job control statement.

On the other hand, you may prefer to take advantage of the CICS dynamic allocation of files. For dynamic allocation, do **not** specify a DD statement for the CSD. CICS then uses the full data set name as specified in the DSNAME parameter of the file resource definition (up to 44 characters), together with the DISP parameter, to allocate the file as part of OPEN processing. This form of dynamic allocation applies equally to files that are defined to be opened explicitly, and those that are to be opened on first reference by an application. For more information about file opening, see Chapter 12, “Defining user files,” on page 111. For information about the parameters that you can code on file resource definitions, see the *CICS Resource Definition Guide*.

The card reader/line printer (CRLP) simulated terminals shown in our sample job stream are defined in the sample TCT (not used in this startup job). See the copy member DFH\$TCTS, in CICSTS31.CICS.SDFHSAMP, for the source statements you need for such devices. For information about defining these devices in a TCT, see the *CICS Resource Definition Guide*.

### **24 Quiescing sequential devices**

For sequential devices, the last entry in the input stream can be CESF GOODNIGHT\ to provide a logical close, and quiesce the device. However, if you close a device in this way, the receive-only status is recorded in the warm keypoint at CICS shutdown. This means that the terminal is still in RECEIVE status in a subsequent warm start, and CICS does not then read the input file.

Note the end-of-data character (the “\” symbol) at the end of each line of the sample.

---

## **Storage requirements for a CICS region**

This section describes considerations about CICS storage requirements. For information about CICS storage requirements, and the effect on CICS performance, see the *CICS Performance Guide*.

When you submit your CICS job, an MVS region is allocated for the execution of CICS. You determine the overall size of the region by coding the REGION parameter, either on the JOB card or on the EXEC PGM=DFHSIP statement. If you specify the REGION parameter on the JOB statement, each step of the job

executes in the requested amount of space. If you specify the REGION parameter on the EXEC statements in a job, each step executes in its own amount of space. Use the EXEC statement REGION parameters when different steps need greatly different amounts of space; for example, when using extra job steps to print auxiliary trace data sets after CICS has shut down (as in the DFHIVPOL installation verification procedure).

The available address space allocated above and below the 16MB line is determined by the value you code on the REGION parameter, but subject to any limits imposed by the IEFUSI exit routine. For details of using the IEFUSI exit routine, see the *OS/390 MVS Installation Exits* manual.

The transaction isolation facility increases the allocation of some virtual storage above the 16MB boundary for CICS regions that are running with transaction isolation active.

If you are running with transaction isolation active, CICS allocates storage for task-lifetime storage in multiples of 1MB for user-key tasks that run above the 16MB boundary. (1MB is the minimum unit of storage allocation above the line for the EUDSA when transaction isolation is active.) However, although storage is allocated in multiples of 1MB above the 16MB boundary, MVS paging activity affects only the storage that is actually used (referenced), and unused parts of the 1MB allocation are not paged.

If you are running without transaction isolation, CICS allocates user-key task-lifetime storage above 16MB in multiples of 64KB.

The subspace group facility uses more real storage, as MVS creates for each subspace a page and segment table from real storage. The CICS requirement for real storage varies according to the transaction load at any one time. As a guideline, each task in the system requires 9KB of real storage, and this should be multiplied by the number of concurrent tasks that can be in the system at any one time (governed by the MXT system initialization parameter).

However, automatic DSA sizing removes the need for accurate storage estimates, with CICS dynamically changing the size of DSAs as demand requires.

For details of how MVS allocates the storage requested by your REGION parameter, see the *OS/390 MVS JCL Reference* manual. For ease of reference, examples of possible size ranges are given here.

**REGION=0K or 0MB**

MVS gives the job all the available private storage below and above the 16MB line. The resulting size of the region below and above 16MB is unpredictable.

**REGION=>0 and ≤16M**

MVS establishes the specified value as the size of the private area below the 16MB line, and a default extended region size of 32MB. If the region size specified is not available, the job step terminates abnormally.

The amount of private storage available below the line varies from installation to installation, and possibly from IPL to IPL, because of the installation-dependent parameters you use to generate and IPL your MVS system. Typically, the amount of common storage required by MVS is 7MB or more, leaving you with a potential private storage area of less than 9MB.

**REGION=>16M and ≤32M**

MVS gives the job all the storage available below 16MB, the size of which is unpredictable, and a default extended region size of 32MB.

**REGION >32M and ≤2047M**

MVS gives the job all the storage available below 16MB, the size of which is unpredictable, and the extended region size as specified. If the region size specified is not available above 16 megabytes, the job step terminates abnormally.

When calculating the size of your CICS region, allow for the following areas of private storage in the CICS region, above and below the 16MB line:

- Storage that CICS reserves at the start of CICS initialization for exclusive use by the CICS kernel.
- Storage that the CICS storage manager reserves for the dynamic storage areas, which you specify on the system initialization parameters, DSALIM and EDSALIM.
- Storage remaining in the private area, after the kernel and DSA requirements have been allocated, to meet additional CICS storage demands from the operating system, for control blocks and buffers for the various access methods.
- If you are using CICS data tables, the amount of storage left after deducting the CDSA requirements must be enough for the records you want to include in the data tables.

For guidance information about virtual storage management in an MVS environment, see the *OS/390 MVS Initialization and Tuning Guide*.

# Paging requirements should be reviewed when you are defining large STG\_SIZE  
# values.

## Storage protection

CICS releases from CICS/ESA 3.3 onward use the extensions added to ESA/390 storage protection facilities, available under MVS/ESA Version 4 Release 2.2, to prevent CICS code and control blocks from being overwritten accidentally by your own user application programs. This is done by allocating separate storage areas (with separate storage keys) for your user application programs, and for CICS code and control blocks. Access to a storage area is not permitted unless the access key matches the key for that storage area.

The storage allocated for CICS code and control blocks is known as **CICS-key** storage, and the storage allocated for your user application programs is known as **user-key** storage. In addition to CICS-key and user-key storage, CICS can also use **key-0 storage** for separate dynamic storage areas below and above the 16MB boundary called the **read-only** DSAs (RDSA and ERDSA). The ERDSA is used for eligible re-entrant CICS and user application programs link-edited with the RENT and RMODE(ANY) attributes. The RDSA is used for eligible re-entrant CICS and user application programs link-edited with the RENT and RMODE(24) attributes. The allocation of key-0 storage for the read-only DSAs is from the same storage limit as the other DSAs, as specified by the DSALIM and EDSALIM system initialization parameters.

Use of the storage protection facilities are optional. You can enable them by coding options on new storage protection system initialization parameters. Between them, these new parameters enable you to define or control:

- The storage key for the common work area (CWAKEY)

- The storage key for the terminal control table user areas (TCTUAKEY)
- A storage protection global option (STGPROT)
- A read-only program storage key option (RENTPGM)
- A transaction isolation option (TRANISO)

To help you get started, CICS provides DFHSIT\$\$, a default system initialization table. This default table is supplied in the CICSTS31.CICS.SDFHSAMP library in source form, and you can modify this to suit your own requirements. When assembled and link-edited, DFHSIT\$\$ becomes the unsuffixed DFHSIT, which is supplied in pregenerated form in CICSTS31.CICS.SDFHAUTH.

### The common work area

The common work area (CWA) is an area of storage within your CICS region that any user application can access. You determine the size of this work area by means of the WRKAREA system initialization parameter, which allows you to specify sizes up to 3584 bytes. If you omit the WRKAREA parameter, CICS allocates a 512-byte CWA by default. You specify the storage key for the CWA on the CWAKEY parameter.

Because this work area is available to all transactions in a CICS region, you should ensure that the storage key is appropriate to the use of the CWA by all transactions. If there is only one transaction that runs in user key, and which requires **write** access, you must specify user-key storage for the CWA, otherwise it will fail with a storage protection exception (an ASRA abend). CICS obtains user-key storage for the CWA by default, and you must review the use of this storage by all programs before you decide to change it to CICS key.

It is possible that you might want to protect the CWA from being overwritten by applications that should not have write access. In this case, provided all the transactions that legitimately require write access to the CWA run in CICS key, you can specify CICS-key storage for the CWA.

See page “CWAKEY ” on page 182 for details of how to specify the CWAKEY parameter.

### The terminal control table user areas

A terminal control user area (TCTUA) is an optional storage area associated with a terminal control table terminal entry (TCTTE), and is available for application program use.

For VTAM terminals, you specify that you want a TCTUA by means of the USERAREALEN parameter on the TYPETERM resource definition. The USERAREALEN parameter on a typeterm definition determines the TCTUA sizes for all terminals that reference the typeterm definition.

For sequential terminals, definitions are added to the terminal control table (TCT), and sizes are defined by means of the TCTUAL parameter on the DFHTCT TYPE=TERMINAL and TYPE=LINE entries. For information about the TCTUAL parameter, see the *CICS Resource Definition Guide*.

You specify the storage key for the TCTUAs globally for a CICS region by the TCTUAKEY system initialization parameter. By default, CICS obtains user-key storage for all TCTUAs.

You must review the use of TCTUAs in your CICS regions, and specify CICS key only for TCTUAs when you are sure that this is justified. If you specify CICS-key storage for TCTUAs, no user-key applications can write to any TCT user areas.

See page “TCTUAKEY ” on page 249 for details of how to specify the TCTUAKEY parameter.

### **The storage protection global option**

You can control whether your CICS region uses storage protection by specifying the STGPROT parameter. By default, CICS does not use storage protection, and all applications run in the same key as CICS, as in earlier releases.

The default option is suitable for pure terminal-owning regions (TORs) and data-owning regions (DORs) that do not execute user transactions. If you want storage protection in a CICS region, you must specify this on the STGPROT parameter.

See page “STGPROT” on page 242 for details of how to specify the STGPROT parameter.

### **Transaction isolation**

CICS transaction isolation builds on CICS storage protection, enabling user transactions to be protected from one another. You can specify transaction isolation globally for a CICS region on the TRANISO (and STGPROT) system initialization parameter.

In addition to being able to specify the storage and execution key individually for each user transaction, you can specify that CICS is to isolate a transaction’s user-key task-lifetime storage to provide transaction-to-transaction protection. You do this by the ISOLATE option of the TRANSACTION or TRANCLASS resource definition.

For an overview of transaction isolation, and CICS’ use of MVS subspaces, see the *CICS Performance Guide* .

### **The read-only storage override option**

CICS obtains storage for the read-only DSAs (RDSA and ERDSA) from MVS read-only storage. CICS loader automatically loads eligible modules into the RDSA and ERDSA; that is, if they are link-edited with the RENT attribute, and for the ERDSA with RMODE(ANY). If you do not want such modules to be loaded into read-only storage (perhaps because you are using a development aid package that sets break points in your application programs) you can override the selection of read-only storage for the RDSA and ERDSA by specifying NOPROTECT on the RENTPGM system initialization parameter.

**Note:** When you specify RENTPGM=NOPROTECT, CICS still allocates separate read-only DSAs, but obtains CICS key-storage for the RDSA and ERDSA instead of read-only storage.

You are recommended to specify RENTPGM=NOPROTECT for development regions only, and to specify RENTPGM=PROTECT for production CICS regions. See page “RENTPGM” on page 227 for details of how to specify the RENTPGM parameter.

## The dynamic storage areas and associated storage cushions

CICS supports eight dynamic storage areas (DSAs), each with its own “storage cushion”. CICS always allocates eight separate DSAs, even if you are running without storage protection (either because the necessary hardware or MVS support is not available, or because you switch off storage protection by means of the STGPROT parameter). If you are running with STGPROT=NO, CICS allocates the DSAs that are controlled by the STGPROT parameter from CICS-key storage (the same storage key as in earlier releases). However, because the CICS and user DSAs are physically separate, it makes the overwriting of CICS storage less likely, even if the DSAs are all in the same storage key.

The type of storage for the read-only DSAs is controlled by the RENTPGM system initialization parameter.

The eight DSAs are as follows:

<b>CDSA</b>	The CICS DSA, allocated below the 16MB boundary, always from CICS-key storage.
<b>RDSA</b>	The read-only DSA, allocated below the 16MB boundary from either read-only storage or CICS-key storage depending on the RENTPGM parameter.
<b>SDSA</b>	The shared DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.
<b>UDSA</b>	The user DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.
<b>ECDSA</b>	The extended CICS-key DSA, allocated above the 16MB boundary, always from CICS-key storage.
<b>ERDSA</b>	The extended read-only DSA, allocated above the 16MB boundary, either from read-only storage or CICS-key storage, depending on the RENTPGM parameter.
<b>ESDSA</b>	The extended shared DSA, allocated above the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.
<b>EUDSA</b>	The extended user DSA, allocated above the 16MB boundary, from either user-key or CICS-key storage depending on the STGPROT parameter.

Table 31 shows the type of storage allocated according to the system initialization parameters specified.

You specify the overall limits within which CICS can allocate the DSAs by the DSALIM and EDSALIM system initialization parameters (for the DSAs below and above the 16MB boundary respectively). Within these limits, CICS dynamically controls the sizes of the individual DSAs and their associated cushions. Also, you can vary these overall limits dynamically, by using either the CEMT SET SYSTEM command or an EXEC CICS SET SYSTEM command.

Table 31. Controlling the storage key for the dynamic storage areas

Dynamic storage area	STGPROT= NO	STGPROT= YES	RENTPGM= PROTECT	RENTPGM= NOPROTECT
CDSA	CICS key	CICS key	N/A <sup>1</sup>	N/A <sup>1</sup>
RDSA	N/A <sup>2</sup>	N/A <sup>2</sup>	Read-only key-0	CICS key
SDSA	CICS key	User key	N/A <sup>1</sup>	N/A <sup>1</sup>

Table 31. Controlling the storage key for the dynamic storage areas (continued)

Dynamic storage area	STGPROT= NO	STGPROT= YES	RENTPGM= PROTECT	RENTPGM= NOPROTECT
UDSA	CICS key	User key	N/A <sup>1</sup>	N/A <sup>1</sup>
ECDSA	CICS key	CICS key	N/A <sup>1</sup>	N/A <sup>1</sup>
ESDSA	CICS key	User key	N/A <sup>1</sup>	N/A <sup>1</sup>
ERDSA	N/A <sup>2</sup>	N/A <sup>2</sup>	Read-only key-0	CICS key
EUDSA	CICS key	User key	N/A <sup>1</sup>	N/A <sup>1</sup>
<p><b>Note:</b> 1. Not applicable. The RENTPGM option has no effect on these DSAs, for which the storage key is determined only by the STGPROT parameter.</p> <p><b>Note:</b> 2. Not applicable. The STGPROT option has no effect on the read-only DSAs, for which the storage key is determined only by the RENTPGM parameter.</p>				

## The storage cushions

CICS reserves amounts of storage in the dynamic storage areas (DSAs) for use when processing storage stress conditions. Each reserved area, which consists of contiguous virtual storage, is called a “storage cushion.” A storage stress condition occurs when CICS cannot satisfy a GETMAIN request, or can satisfy it only by using some of the cushion storage even when all programs that are eligible for deletion, and not in use, have been deleted. This may lead to a short-on-storage condition, if CICS cannot rectify the stress condition.

CICS dynamically tunes the size of the DSA storage cushions as necessary, within the limits set by the DSALIM and EDSALIM system initialization parameters. However, if the amount of storage available for the storage cushions becomes too small, an SOS condition can still occur.

**Effects:** In a storage stress condition, the cushion mechanism can avert a storage deadlock condition. This prevents CICS taking on additional work by stopping most of the soliciting for new input messages. For information on the effects of stress conditions, see the *CICS Performance Guide*.

CICS sets the storage stress condition if:

- After any successful GETMAIN, the number of unallocated dynamic storage pages remaining is less than the cushion size. This is shown in the storage statistics as “Times cushion released”.
- CICS cannot satisfy an unconditional GETMAIN because there is no contiguous area large enough for it. This is shown in the storage statistics as “Times request suspended”.

When a storage stress situation exists, the loader domain attempts to alleviate it by releasing the main storage for programs with no current user. If this fails, a short-on-storage condition is indicated, and a message is issued at the console.

While the SOS condition is set, acquisition of new input message areas is prevented, and all ATTACH requests from CICS system modules are deferred.

## Recommendations

To help CICS optimize its use of the DSAs and their storage cushions, you are recommended to:

- Avoid using large GETMAIN requests.

The storage cushion is a contiguous block of storage of fixed size, and therefore may be able to satisfy a request for a large contiguous block of storage.

- Minimize the number of resident programs.

If storage cushion releases occur frequently, you need to find out why. Reduce the maximum number of user tasks (the MXT system initialization parameter) to reduce the number of tasks using main storage. You may need either to alter your application programs so that they do not issue large GETMAINS.

Only transactions defined as SPURGE(YES) and with a DTIMOUT value can be purged during an SOS condition if they have been waiting for storage for longer than the DTIMOUT value. If such transactions are too few and if storage becomes totally deadlocked, the system can stall.

### **How implemented**

CICS allocates the initial size of the storage cushions for the DSAs from the overall storage limits defined by the DSALIM and EDSALIM system initialization parameters. CICS dynamically tunes the sizes of the DSAs and their storage cushions within these limits.

For descriptions of the DSALIM and EDSALIM system initialization parameters, see page “DSALIM” on page 185 and “EDSALIM” on page 189 respectively.

You can change the overall storage limits while CICS is running by means of a CEMT SET SYSTEM command or an EXEC CICS SET command.

### **How monitored**

Storage stress conditions are notified in the storage statistics (“Times cushion released” and “Times request suspended”). A storage stress condition may not cause an SOS condition; CICS may be able to alleviate the condition. However, storage stress conditions are costly, and should be avoided.

The SOS condition is notified in the dynamic storage area statistics (“times went short on storage”), and is made apparent to the terminal user by external effects such as ceasing of polling and transaction initiation, and prolonged response times. In addition, a message is displayed on the operating system console when the short-on-storage (SOS) indication is detected. The SOS message, DFHSM0131 or DFHSM0133, indicates that:

- The amount of free space in a dynamic storage area is less than needed, and the associated DSA cannot be enlarged further (because the DSA limit has been reached).
- There are currently suspended GETMAIN requests waiting for large enough areas of contiguous storage to become available.

If there is insufficient storage in the relevant DSA limit to satisfy a GETMAIN request, the request is either queued (for unconditional requests) or a return code is given (for conditional requests).

### **Coding conventions for DSA limits**

You can specify the size of the DSA limits as:

- A number of bytes
- A whole number of kilobytes
- A whole number of megabytes

Use the letter K or M as a suffix to indicate whether the value represents a whole number of kilobytes, or a number of megabytes. For example, 2MB can be coded as either 2048K or 2M. (1KB = 1024 bytes; 1MB = 1024KB = 1048576 bytes.)

If the value you specify is not a multiple of 256KB for DSALIM, or 1MB for EDSALIM, CICS rounds up the value to the next multiple.

You cannot specify fractions of megabytes: you must code sizes in bytes or kilobytes. Some examples are shown in Table 32:

Table 32. Examples of DSA limit values in bytes, kilobytes and megabytes

Coded as:					
bytes	2097152	3145788	3670016	4194304	4718592
kilobytes	2048K	3072K	3584K	4096K	4608K
megabytes	2M	3M	-	4M	-

For information about estimating the size of the dynamic storage areas, see the *CICS Performance Guide*.

---

## The sample statistics program, DFH0STAT

You can use the statistics sample program, DFH0STAT, to help you determine and adjust values needed for CICS storage parameters, for example the size of the DSALIM and EDSALIM system initialization parameters. The program produces a report showing critical system parameters from the CICS Dispatcher, an analysis of the CICS Storage Manager and Loader statistics, and an overview of the MVS storage in use. The program demonstrates the use of EXEC CICS INQUIRE and EXEC CICS COLLECT STATISTICS commands to produce an analysis of your CICS. You can use the sample program as provided or modify it to suit your needs.

For more information on the statistics sample program, DFH0STAT, see the *CICS Performance Guide*.

---

## A sample CICS startup procedure

As an alternative to submitting a batch job to the MVS internal reader, you can use the MVS START command to start CICS as a started task. Using this method, your startup job stream must be coded according to the rules for coding procedures, and the procedure must be installed in an MVS procedure library.

Note that if you intend to start CICS with the START command you must either:

- Give the MVS started task procedure a name different from the subsystem name in IEFSSNaa (default 'CICS'), or
- Issue the start command with the parameter SUB=JES2 or SUB=JES3 as appropriate.

You can use the following form of the MVS START command to start a job from the console:

```
S|START procname[.identifier] [,SUB=subsystemname] [,keyword=option  
[,keyword=option] . . .]
```

**procname**

The name of the cataloged procedure that defines the job to be started.

**identifier**

The name you choose to identify the task.

**SUB=subsystemname**

The name of the subsystem that is to select the job for processing. If you omit this parameter, the primary job entry subsystem is used.

**keyword=option**

Any appropriate keyword to override the corresponding parameter in the procedure. You can use this parameter to override symbolic parameters defined in the cataloged procedure.

For guidance information about the complete syntax of the START command, and all the keywords and options you can use, see the *OS/390 MVS System Commands* manual.

To start CICS, you only need to code **procname.identifier,keyword(s)=option**.

For example:

```
START DFHSTART.CICSA,SIP=T,REGNAME1=IDA,REGNAM2=IDA
```

In this example of the MVS START command:

- DFHSTART is the name of the CICS-supplied cataloged startup procedure.
- CICS is being started with a task ID of CICSA.
- SIP is the suffix of the DFH\$SIPx member in the SYSIN data set, CICSTS31.CICS.SYSIN, to be used by this CICS region.
- REGNAM1 and REGNAM2 are qualifiers added to the CICS system data sets specified in the procedure (for example, CICSTS31.CICS.CICSHTH1.DFHTEMP) to identify uniquely the data sets for this CICS region. REGNAM1 is set to the same value as REGNAM2 for an XRF active CICS region or an MRO region.

For information about the DFHSTART procedure, see the *CICS Transaction Server for z/OS Installation Guide*.

If you are running CICS with RACF, you must associate the cataloged procedure name with a suitably authorized RACF user through the RACF table, ICHRIN03. For information about this association, see the *CICS RACF Security Guide*.



---

## Chapter 22. Preparing CICS for using debugging tools

Before application programmers can use certain debugging tools with CICS, you will need to perform some system definition tasks. This chapter describes those tasks.

It contains the following topics:

- “Preparing your CICS region for debugging”

---

### Preparing your CICS region for debugging

Before application programmers can use certain debugging tools with CICS, you must configure your CICS region accordingly. This topic describes preparing your CICS region for debugging application programs using the following tools:

- Debug Tool, for compiled language application programs (programs written in COBOL, PL/I, C, C++), and for Language Environment-enabled Assembler subroutines).
- Remote debugging tools (for compiled language application programs, Language Environment-enabled Assembler Subroutines, and Java programs). Note that for compiled language programs, and Assembler subroutines, Debug Tool is used as the debugging server.

This topic does not apply to other debugging tools, such as the CICS Execution Diagnostic Facility (CEDF).

Since you will need to restart the region, it is advisable to plan which regions will be used for debugging applications.

- It is unlikely that application programs will be debugged in your production regions, especially if the applications are well established and known to be reliable; it is more likely that debugging will take place in regions which are used to develop and test new applications.

To prepare your CICS region for debugging:

1. If you plan to use the region for debugging compiled language programs, include the Debug Tool library SEQAMOD in the DFHRPL concatenation in your CICS startup JCL.

For more information, see “Using the sample startup job stream” on page 338.

2. Create the *debugging profile data sets*.

For more information, see Chapter 17, “Setting up the debugging profiles data sets,” on page 143.

3. Include the resource definition for the debugging profile file in a resource definition list that is named in the GRPLIST system initialization parameter.
4. Optionally, specify the following value for the DEBUGTOOL system initialization parameter:

```
DEBUGTOOL=YES
```

If you do not specify DEBUGTOOL=YES, you can enable the region for debugging when it is running:

- To enable the region for debugging from a program, use the EXEC CICS SET SYSTEM DEBUGTOOL command
- To enable the region for debugging from the master terminal transaction, use the CEMT SET SYSTEM DEBUGTOOL command

Enabling the region for debugging when it is running is recommended for regions which are not normally used for debugging. When debugging is complete, you can disable the region for debugging, using the same commands.

5. Define and install Debug Tool's resource definitions. They are located in member EQACCSO in Debug Tool's SEQASAMP data set. For more information, see the *Debug Tool for z/OS and OS/390 User's Guide*.

---

## Part 4. Initializing CICS data sharing servers

This section of the book describes how to initialize CICS data sharing servers to support:

- Temporary storage data sharing
- Coupling facility data tables
- Named counters.

It also describes how to define and start AXM system services, on which the data sharing servers depend. The contents of this Part of the book are:

- Chapter 23, “Authorized cross memory (AXM) system services,” on page 367 describes how to define and start AXM system services
- Chapter 24, “Setting up and running a temporary storage server,” on page 369 describes how to set up and run a CICS temporary storage data sharing server
- Chapter 25, “Setting up and running a coupling facility data table server,” on page 383 describes how to set up and run a CICS coupling facility data table server
- Chapter 27, “Coupling facility server operations,” on page 423 gives an overview of the operations which are common to all three CICS coupling facility servers.
- Chapter 26, “Setting up and running a named counter server,” on page 405 describes how to set up and run a CICS named counter server.



---

## Chapter 23. Authorized cross memory (AXM) system services

This chapter describes how to define and start AXM system services, which are required by the CICS data sharing servers. If you plan to use any of the following facilities, you must first ensure that AXM system services are started:

- Temporary storage data sharing
- Coupling facility data table
- Named counters.

---

### The authorized cross-memory (AXM) server environment

The execution environment for CICS data sharing server regions is provided by a run-time environment package called the authorized cross-memory (AXM) server environment. To establish AXM cross-memory connections for an MVS image, initialize the AXM system services by defining an MVS subsystem called AXM. The AXM subsystem initialization routine, AXMSI, sets up the appropriate definitions from the master scheduler region. Note that AXM uses the subsystem definition only as a means of scheduling AXM initialization in the master scheduler address space. The MVS subsystem interface for AXM is not activated or used.

To define the AXM subsystem statically, the normal method, add an entry with the required parameters to the IEFSSNxx member of SYS1.PARMLIB, as follows:

```
SUBSYS SUBNAME(AXM) INITRTN(AXMSI)
```

Defining AXM in the **IEFSSNxx** member ensures that AXM system services automatically become available when you IPL MVS.

To avoid the need to wait for an IPL when AXM modules are first installed, you can also initialize AXM system services by defining the subsystem dynamically, as follows:

```
SETSSI ADD,SUBNAME=AXM,INITRTN=AXMSI
```

If initialization of the AXM subsystem fails for any reason, (for example, because of an error in the command, or because AXMSI is not in a linklist library) MVS does not allow another attempt because the subsystem is then already defined. In this case, you should use a different subsystem name, such as AXM1, because AXM does not rely on a specific subsystem name. If you start AXM successfully the first time, further attempts are ignored.



---

## Chapter 24. Setting up and running a temporary storage server

This chapter describes how to start up temporary storage servers, and provides information about the following:

- Overview of the temporary storage data sharing server
- Defining TS server regions
- “Queue server automatic ALTER processing” on page 377
- “Shared TS queue server commands” on page 378
- “Unloading and reloading queue pools” on page 380

**Note:** See “Defining temporary storage pools for temporary storage data sharing” on page 38 for guidance on defining the sizes of temporary storage pools.

---

### Overview of the temporary storage data sharing server

Access to a TS pool by CICS transactions running in an AOR is through a TS data sharing server that supports a named pool. In each MVS image in the sysplex, you need one TS server for each pool defined in a coupling facility which can be accessed from that MVS image. All TS pool access is performed by cross-memory calls to the TS server for the named pool.

An AOR can access more than one TS server concurrently. This multiserver access is required if you create multiple pools, because each TS server provides access to only one pool of TS queues.

CICS maps temporary storage requests to a TS server using the `POOLNAME` attribute of a `TSMODEL` resource definition, if one exists. Alternatively, if you are using a temporary storage table (TST) instead of `TSMODEL` resource definitions, CICS maps temporary storage requests to a TS server by means of the `SYSIDNT` option on the `TST TYPE=SHARED` macro.

The methods for specifying a TS pool make it easy to migrate queues from a QOR to a TS data sharing pool. You can use the TS global user exit, `XTSREQ`, to modify the `SYSID` on a TS request so that it references a TS data sharing pool.

Figure 47 on page 370 illustrates a parallel sysplex with three CICS AORs linked to the temporary storage server address space(s).

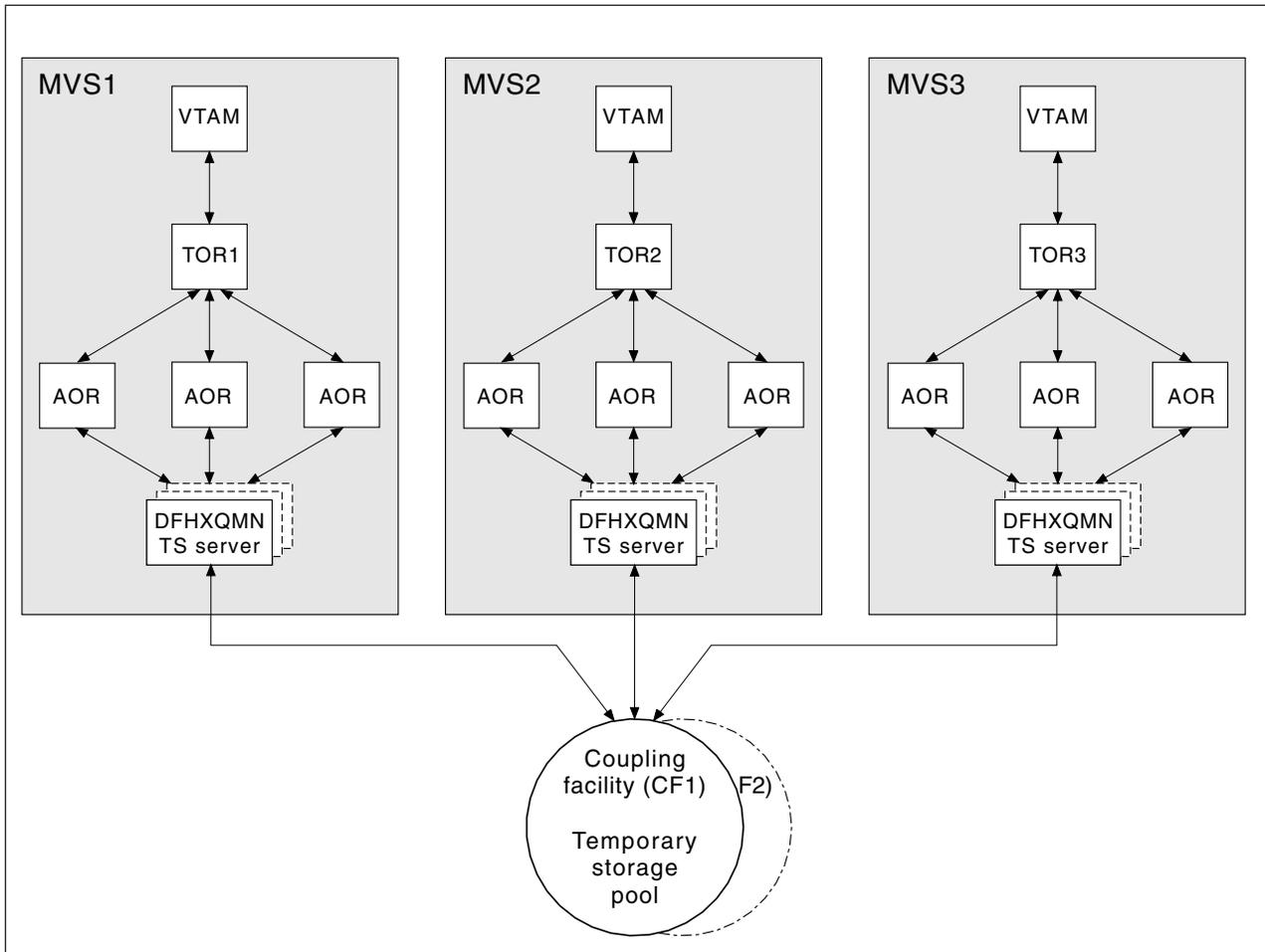


Figure 47. Conceptual view of a Parallel Sysplex with TS data sharing.

## Security

The server must be authorized to access the coupling facility list structure in which the temporary storage pool is defined; XES checks this. The server must also be authorized to act as a temporary storage server; AXM checks this. For information on how to define the necessary authorizations see the *CICS RACF Security Guide*.

## Defining TS server regions

You must ensure that the TS server region is activated before the CICS region needs it. A shared TS pool consists of an XES list structure, which is accessed through a cross-memory queue server region. A shared TS pool is started in an MVS image by starting up a queue server region for that pool as either a batch job or a started task. This invokes the queue server region program, DFHXQMN, which resides in an APF-authorized library.

DFHXQMN requires some startup parameters, of which the TS pool name is mandatory. A SYSPRINT DD statement is required for the print file, and a SYSIN DD statement for the server parameters. Optional parameters include the maximum number of queues to be supported in the pool and the number of buffers the server is to allocate.

You can specify the DFHXQMN initialization parameters either in a SYSIN data set defined in the JCL, or in the PARM parameter on the EXEC statement.

## Sample startup job for a TS server

Figure 48 shows an example of the JCL you might use to start a TS server.

```
//PRODTSQ1 JOB ...
//TSSERVER EXEC PGM=DFHXQMN,REGION=64M,TIME=NOLIMIT Start TS data sharing server
//STEPLIB DD DSN=CICSxxx.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=* Messages and statistics
//SYSIN DD *
POOLNAME=PRODTSQ1 Pool name
MAXQUEUES=5000 Allow up to 5000 large queues
BUFFERS=1000 1000 buffers (32K each, 32M total)
/*
```

**Note:** You are recommended to specify TIME=NOLIMIT. The server task remains in a wait during most normal processing, because server processing is performed under the client CICS region TCB. If you omit the TIME subparameter, your server job could fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRMxx member of SYS1.PARMLIB.

Figure 48. Sample startup job for a TS queue server

## Queue server REGION parameter

The queue server REGION parameter JCL needs to specify at least enough virtual storage for the specified number of buffers plus the storage used to process queue requests. Each buffer occupies a little more than 32K bytes, and each connected CICS region can have up to ten queue requests active at a time, each using 5K to 10K bytes, so to be safe the REGION size should allow at least 32K per buffer and 100K for each connected CICS region, plus a margin of about 10% for other storage areas.

During server initialization, the server acquires all of the available storage above the 16M line, as determined by the REGION size, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below the line for use in routines which require 24-bit addressable storage, for example sequential file read and write routines.

After server initialization, AXM page allocation services are used to manage server region storage. Server statistics indicate how much storage is actually allocated and used within the storage areas above and below the 16M line, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate, so it is best to ensure that the REGION size is large enough to eliminate the risk.

## TS queue server parameters

Parameters are specified in the form **KEYWORD=value**. You can optionally specify keywords in mixed case to improve readability.

If you specify more than one parameter in the **PARM** field, or on the same **SYSIN** input line, the parameters must be separated by commas. Any text following one or more spaces is taken as a descriptive comment. Any parameter line starting with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as an abbreviation. The standard form of each keyword is generally the longest form of the first word shown.

The main parameters used are listed on the server print file during start-up.

The following parameters are all valid as initialization parameters (in the **SYSIN** file, or the **PARM** field), and some can be modified by the server **SET** command.

You can display any parameter with the server **DISPLAY** command. Display the values of all parameters using **DISPLAY ALLPARMS**.

Specify the following keywords to give combined lists of information:

- **PARMS** for main parameter values
- **STATS** for all available statistics
- **INIT** to select parameters and statistics whose values are usually displayed when initialization is complete

## Primary parameters

These parameters are usually specified for all servers:

**POOLNAME**=*pool\_name*

specifies the name, of 1 to 8 characters, of the queue pool used to form the server name and the name of the coupling facility list structure **DFHXQLS\_poolname**. This parameter is valid only at initialization, and must always be specified.

This keyword can also be coded as **POOL**.

**BUFFERS**={**100**|**number**}

specifies the number of queue buffers to allocate for the server address space.

A queue index buffer holds a queue index entry plus up to 32K of queue data (for a small queue). When a **READ** or **WRITE** request completes the queue index information is retained in the buffer. This can avoid the need to reread the queue index if the same queue is referenced from the same **MVS** image before the buffer has been reused. If no buffer is available at the time of a request, the request is made to wait until one becomes free.

The number of buffers should preferably be at least ten for each **CICS** region that can connect to the server in this **MVS** image. This avoids the risk of buffer waits. Additional buffers may be used to reduce the number of coupling facility accesses by keeping recently used queue index entries in storage. In particular, if the current version of a queue index entry is in storage at the time a queue item is read, the request requires only one coupling facility access instead of two. If the current version of a queue index entry is in storage when a second or subsequent item is written to the same queue, the request requires only one coupling facility access instead of three.

It is not worth defining extra buffers beyond the point where this might cause MVS paging, as it is more efficient to reread the index entry than to page in the buffer from auxiliary storage. This parameter is valid only at initialization.

The valid range is from 1 to 999999.

This keyword can also be coded as **BUF**.

**FUNCTION={SERVER|UNLOAD|RELOAD}**

Information about this parameter is given in “Unloading and reloading queue pools” on page 380.

**STATSOPTIONS={NONE|SMF|PRINT|BOTH}**

specifies the statistics options that determine whether interval statistics are produced and whether statistics are sent to SMF, the print file, or both.

This keyword can also be coded as **STATSOPT**.

**ENDOFDAY={00:00|hhmm}**

specifies the time when end of day statistics are to be collected and reset. If statistics options specify NONE, end of day statistics are written to the print file. The valid range is from 00:00 to 24:00.

This keyword can also be coded as **EOD**.

**STATSINTERVAL={3:00|hhmm}**

specifies the statistics interval, within the range of 1 minute to 24 hours. It is ignored if STATSOPTIONS=NONE.

The valid range is from 00:01 to 24:00 (although it may be specified in seconds).

This keyword can also be coded as **STATSINT**.

## Automatic restart manager (ARM) parameters

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails.

Use the following parameters to override default processing for the automatic restart manager:

**ARMELEMENTNAME=elementname**

specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 \$ # @ and the underscore symbol (\_).

The default identifier is of the form DFHXQnn\_poolname, where XQ represents the server type, nn is the &SYSCClone value for the system (which can be either one or two characters), and poolname is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

**ARMELEMENTTYPE=elementtype**

specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 \$ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

## List structure parameters

The following parameters specify list structure attributes and are used only for initial allocation of the pool list structure, which occurs the first time a server is started for the pool:

**POOLSIZE**={0|number\_of\_bytes{K|M|G}}

specifies the maximum amount of storage to be allocated for the list structure, expressed as kilobytes in the form *nK*, or megabytes in the form *nM*, or gigabytes in the form *nG*.

This takes effect when the list structure is being created with a specified value of less than that specified for the list structure in the CFRM policy.

The default value 0 specifies that no maximum limit is to be applied other than that specified in the CFRM policy. A non-zero value is generally rounded up by MVS to the next multiple of 256K.

The valid range is from 0 to 2G.

**MAXQUEUES**={1000|number}

specifies the maximum number of data lists to be reserved when the structure is allocated, which determines the maximum number of large queues that can be stored in the structure. Note that this parameter also determines how many list headers are defined when the structure is created. Although you should take care to specify a large enough number to handle large queues, specifying an excessively large number will use up an unnecessary amount of coupling facility storage for preallocated list headers.

This number cannot be changed without reallocating the structure. Therefore, if the structure is being allocated at less than its maximum size, the value here should be based on the maximum possible size of the structure rather than its initial size.

The valid range is from 1 to 999999.

This keyword can also be coded as **MAXQ**.

For information about defining list structures, see “Defining temporary storage pools for temporary storage data sharing” on page 38.

## Debug trace parameters

These parameters are used only for intensive debug tracing.

Note that using these options in a production environment may significantly impact performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests may be lost if they are generated faster than the trace print subtask can print them. In such cases, the trace indicates only how many messages were lost.

**TRACECF**={OFF|ON}

specifies the coupling facility interface debug trace options, OFF or ON. This option produces trace messages on the print file indicating the main parameters to the coupling facility request interface and the result from the IXLLIST macro.

This keyword can also be coded as **CFTR** or **CFTRACE**.

**TRACERQ={OFF|ON}**

specifies the queue request debug trace options, OFF or ON. This option produces trace messages on the print file indicating the main parameters on entry to the shared queue request or shared queue inquire interface and the results on exit.

This keyword can also be coded as **RQTR** or **RQTRACE**.

## Tuning parameters

The following parameters are provided for tuning purposes. They are normally allowed to assume their default values:

**ELEMENTSIZE={256|number}**

specifies the element size for structure space, which must be a power of 2. For current coupling facility implementations there is no known reason to specify other than the default value of 256.

This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is 256 to 4096.

This keyword can also be coded as **ELEMSIZE**.

**ELEMENTRATIO={1|number}**

specifies the element side of the entry/element ratio when structure is first allocated. This determines the proportion of the structure space initially set aside for data elements.

The ideal value for this ratio results from the average size of data for each entry being divided by the element size. However, the server automatically adjusts the ratio according to the actual entry and element usage.

This parameter is valid only at server initialization, and is used only when the structure is first allocated.

The valid range is from 1 to 255.

This keyword can also be coded as **ELEMTRATIO**.

**ENTRYRATIO={1|number}**

specifies the entry side of the entry/element ratio when the structure is first allocated. It determines the proportion of structure space initially to be set aside for list entry controls.

It is not essential to specify this parameter because the server automatically adjusts the ratio based on actual usage to improve space utilization if necessary.

This parameter is valid only at server initialization and is used only when the structure is first allocated.

The valid range is from 1 to 255.

**LASTUSEDINTERVAL={00:10|hmm}**

specifies how often the last used time for large queues is to be updated.

For small queues, the last used time is updated on every reference. For large queues, updating the last used time requires an extra coupling facility access, so that it is done only if the queue has not previously been accessed within this interval of the current time. This means that the last used time interval returned by INQUIRE can be greater than the true value by an amount up to the value specified on this parameter. As the main purpose of the last used time specification is to determine whether the queue is obsolete, an interval of a few minutes should be sufficient.

The valid range is from 00:00 to 24:00 (although it may be specified in seconds).

This keyword can also be coded as **LASTUSEDINT**.

**SMALLQUEUEITEMS={9999|number}**

specifies the maximum number of items that can be stored in the small queue format in the queue index entry data area. This parameter can force a queue to be converted to the large queue format if it has a large number of small items. It can be more efficient to write the items separately than to rewrite the whole small queue data area each time.

The valid range is from 1 to 32767.

**SMALLQUEUESIZE={32K|number}**

specifies the maximum data size for a small queue including the two-byte length prefix on each data item. Any queue exceeding the maximum size, when writing the second or subsequent item to a queue, is converted to the large queue format.

This parameter can force queues to be converted to the large queue format at a smaller size than 32K. This is to prevent large amounts of data being written to the small queue format. Performance improvements can result on systems where asynchronous coupling facility processing causes contention for hardware resources. On most systems, however, it is probably more efficient to defer conversion until the maximum size of 32K is reached.

The valid range is from 4096 to 32768.

## Warning parameters

These parameters modify the threshold at which warning messages and automatic ALTER actions occur when the structure becomes nearly full:

**ELEMENTWARN={80|number}**

specifies the percentage of elements in use at which warnings and automatic ALTER actions should be first triggered.

The valid range is from 1 to 100.

This keyword can also be coded as **ELEMWARN**.

**ELEMENTWARNINC={5|number}**

specifies the percentage increase (or decrease) of elements in use before the next warning should be triggered (reduced to 1 when next increase would otherwise reach 100). Additional messages are issued as the number of elements in use changes. The messages stop when the number of elements in use falls at least by this percentage below the initial warning level.

The valid range is from 1 to 100.

This keyword can also be coded as **ELEMWARNINC**.

**ENTRYWARN={80|number}**

specifies the percentage of entries in use at which warnings and automatic ALTER actions should be first triggered.

The valid range is from 1 to 100.

**ENTRYWARNINC={5|number}**

specifies the percentage increase (or decrease) of entries in use before next warning should be triggered (reduced to 1 when next increase would otherwise reach 100). Additional messages are issued as the number of elements change.

The messages stop when the number of entries in use falls at by least the specified percentage below the initial warning level.

The valid range is from 1 to 100.

## Automatic ALTER parameters

Define the following parameters to modify the conditions under which the server attempts an automatic ALTER action when the structure becomes nearly full. For details of the queue server automatic ALTER process, see “Queue server automatic ALTER processing.”

**ALTERELEMMIN**={100|number}

specifies the minimum number of excess elements that must be present for an automatic ALTER to be issued to convert those elements to entries.

The valid range is from 1 to 999999999.

**ALTERELEMPC**={1|number}

specifies the minimum percentage of excess elements that must be present for an automatic ALTER to be issued to increase the proportion of entries.

The valid range is from 0 to 100.

**ALTERENTRYMIN**={100|number}

specifies the minimum number of excess entries that must be present for an automatic ALTER to be issued to convert those entries to elements.

The valid range is from 0 to 999999999.

**ALTERENTRYPC**={1|number}

specifies the minimum percentage of excess entries which must be present for an automatic ALTER to be issued to increase the proportion of elements.

The valid range is from 0 to 100.

**ALTERMININTERVAL**={00:10|hhmm}

specifies the minimum time interval to be left between automatic ALTER attempts when the structure is nearly full (above the element or entry warning level).

The valid range is from 00:00 to 24:00.

This keyword can also be coded as **ALTERMININT**.

---

## Queue server automatic ALTER processing

The queue server monitors the total number of elements and entries in use in the structure, using information returned by the coupling facility on every request. When the numbers in use exceed the specified thresholds, a warning message, DFHXQ0411 or DFHXQ0412, is issued, and is repeated each time the number in use increases beyond further thresholds.

Each time the warning is issued, the server tests whether an automatic ALTER for the entry to element ratio should be performed. The test is done by calculating how many excess elements or entries will be left when the other runs out completely. This is based on the ratio between the current numbers of elements and entries actually in use.

An IXLALTER request is issued to alter the entry to element ratio to the actual current ratio between the number of entries and elements in use if:

- The number of excess elements or entries exceeds the number specified in the ALTERELEMMIN or ALTERENTRYMIN parameter, and
- The same number expressed as a percentage of the total exceeds the value specified in the ALTERELEMPC or ALTERENTRYPC parameter

Only one ALTER request may be active at a time for a given structure. If the ALTER process is started by one server, the ALTER of another server will be rejected. However, the system automatically notifies all servers when the ALTER completes, giving the new numbers of elements and entries so that each server can update its own status information.

A further ALTER attempt is suppressed until at least the minimum ALTER interval (specified by the ALTERMININTERVAL parameter) has elapsed, if:

- Any form of ALTER has already been used recently (by any server or by an operator SETXCF ALTER command), and
- The structure space usage has remained above warning levels since the previous attempt.

---

## Shared TS queue server commands

Commands can be issued to control a queue server using the MVS MODIFY (F) command specifying the job or started task name of the server region. The general form of a queue server command is as follows:

```
F server,cmd parameter,parameter... comments
```

The MVS STOP command is equivalent to issuing the server command STOP using the MVS MODIFY command.

The queue server supports the following commands:

### **SET keyword=value**

changes one or more server parameter values. This applies to all parameters other than those indicated as being for initialization only. The command can be abbreviated to T, as for the MVS SET command.

### **DISPLAY keyword**

displays one or more parameter values, or statistics summary information, on the console. The valid parameter keywords for DISPLAY and PRINT are described later in this section. The command can be abbreviated to D, as for the MVS DISPLAY command.

### **PRINT keyword**

produces the same output as DISPLAY but only on the print file.

### **STOP**

terminates the server, waiting for any active connections to terminate first, and preventing any new connections.

The command can be abbreviated to P, as for the MVS STOP command.

### **CANCEL {RESTART={NO|YES}}**

Terminate the server immediately. You can specify whether automatic restart should be requested.

For information about CANCEL RESTART see “The CANCEL command options” on page 380.

The server also responds to XES events such as an operator SETXCF command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server CANCEL command to close itself down immediately.

## DISPLAY and PRINT keywords

DISPLAY or PRINT can display the values of any initialization parameters plus the following additional information:

### ARMREGISTERED

Shows whether ARM registration was successful (YES or NO).

### CONNECTIONS

List of the jobnames and applids for the regions currently connected to this server. This keyword can also be coded as **CONN**.

### Statistics summaries:

#### BUFSTATS

Queue index buffer pool statistics.

This keyword can also be coded as **BUFST**.

#### CFSTATS

Coupling facility interface I/O and response statistics.

This keyword can also be coded as **CFST** or **STATSCF**.

#### POOLSTATS

Usage statistics for the pool list structure as a whole.

This keyword can also be coded as **POOLST**.

#### STORAGESTATS

Main storage allocation statistics for the server address space.

This keyword can also be coded as **STGST**, **STGSTATS**, or **STORAGEST**.

### Keywords representing combined lists of information

#### PARAMETERS

Main parameter values.

The keyword can also be coded **PARM**, **PARMS**, or **PARAM**.

#### ALLPARAMETERS

All parameter values.

This keyword can also be coded as **ALLPARMS**.

#### STATISTICS

All available statistics.

This keyword can also be coded as **STAT** or **STATS**.

#### INITIALIZED

Selected parameters and statistics whose values are usually displayed when initialization is complete.

This keyword can also be coded as **INIT**.

#### ARM

Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

## The CANCEL command options

You can use the CANCEL command to request automatic restart by specifying the following parameter:

**RESTART={NO|YES}**

Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

You can also enter RESTART on its own for RESTART=YES, NORESTART for RESTART=NO.

---

## Unloading and reloading queue pools

# The contents of a queue pool can be unloaded to a sequential file and  
# subsequently reloaded by running the server program using the FUNCTION  
# parameter. This can be used to preserve the queue pool across planned coupling  
# facility maintenance, or to move the queue pool to a different coupling facility.  
# Alternatively, you can use the system-managed rebuild facility to dynamically move  
# a structure to another coupling facility connected to the same system. This allows  
# servers to remain active, but temporarily suspends processing while the rebuild is in  
# progress. For more information, see “System-managed list structure rebuild” on  
# page 431.

**FUNCTION={UNLOAD|RELOAD}**

requests the server to perform the special functions UNLOAD or RELOAD. When the unload or reload processing has completed (normally or abnormally) the server program terminates.

If this parameter is omitted, the server program initializes the cross-memory queue server environment.

When UNLOAD or RELOAD is specified, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the attempt to unload or reload is rejected. Similarly, if a normal server attempts to start up while an unload or reload function is in progress, the attempt is rejected because shared access to the structure is not available.

All normal server parameters can be specified on UNLOAD and RELOAD, but many of these, such as the number of queue buffers, are ignored because they do not apply to unload or reload processing.

The UNLOAD function requires a DD statement for file name DFHXQUL describing the sequential data set to which the queue pool is to be unloaded. The format of the unloaded file is:

RECFM=F,LRECL=4096,BLKSIZE=4096.

An upper limit for the total size of the data set in bytes can be estimated from the pool usage statistics produced by the server. The total data size in bytes is obtained by multiplying the number of elements in use by the element size (usually 256), and for each queue there is also some control information which typically occupies fewer than 100 bytes per queue. The size is normally smaller than this because unused space in data elements is not included in the unloaded file. See Figure 49 for an example of UNLOAD JCL.

```
//UNLDTSQ1 JOB ...
//TSUNLOAD EXEC PGM=DFHXQMN          CICS TS queue server program
//STEPLIB DD DSN=CICSxxx.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD SYSOUT=*                Options, messages and statistics
//DFHXQUL DD DSN=TSQ1.UNLOADED.QPOOL,  Unloaded queue pool
//          DISP=(NEW,CATLG),
//          SPACE=(4096,(10000,1000))  Estimated size in 4K blocks
//SYSIN DD *
FUNCTION=UNLOAD                      Function to be performed is UNLOAD
POOLNAME=PRODTSQ1                   Pool name
/*
```

Figure 49. Example of UNLOAD JCL

The RELOAD function requires a DD statement for file name DFHXQUL describing the sequential data set from which the queue pool is to be reloaded. The structure is allocated if necessary during reloading, in which case the same server parameters may be used to control structure attributes as for normal server execution. The RELOAD process bypasses any queues that are already found in the queue pool, because, for example, the structure was too small and the reload job had to be restarted after using ALTER to increase the size.

Note that when a pool is nearly full (with less than about 5% free entries and elements) there is no guarantee that it can be unloaded and reloaded into a structure of exactly the same size. The amount of space available is affected by the current ratio of entries to elements, which can only be controlled approximately by the automatic ALTER process.

If the structure reaches the warning level during reloading, the automatic ALTER process attempts to adjust the entry to element ratio, and the reload process will automatically wait for the ALTER to complete if no space is available while an ALTER is still in progress.

If RELOAD fails because it runs out of space, the resulting messages include the numbers of queues reloaded and blocks read up to the time of the failure. Compare these values with those in the messages from the original UNLOAD to determine how many more queues and how much more data remained to be loaded. See Figure 50 on page 382 for an example of RELOAD JCL.

```

//RELDTSQ1 JOB ...
//TSRELOAD EXEC PGM=DFHXQMN          CICS TS queue server program
//STEPLIB DD DSN=CICSxxx.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD SYSOUT=*                Options, messages and statistics
//DFHXQRL DD DSN=TSQ1.UNLOADED.QPOOL,DISP=OLD Unloaded queue pool
//SYSIN DD *
FUNCTION=RELOAD                      Function to be performed is RELOAD
POOLNAME=PRODTSQ1                    Pool name
POOLSIZE=50M                         Increased pool size
MAXQUEUES=10000                      Increased number of big queues
/*

```

Figure 50. Example of RELOAD JCL

---

## Chapter 25. Setting up and running a coupling facility data table server

This chapter describes how to start up a coupling facility data table server, and provides information about the following:

- “Overview of a coupling facility data table server”
- “Defining and starting a coupling facility data table server region” on page 384
- “Controlling coupling facility data table server regions” on page 396
- “Deleting or emptying coupling facility data table pools” on page 402
- “Unloading and reloading coupling facility data table pools” on page 402.

**Note:** Before you can start a server for named coupling facility data table pool, first define the coupling facility structure to be used for the pool. See “Defining a coupling facility data table pool” on page 126 for information about defining a coupling facility list structure for a CFDT.

---

### Overview of a coupling facility data table server

CICS coupling facility data tables is designed to provide rapid sharing of working data within a sysplex, with update integrity. The data is held in a table that is similar in many ways to a shared user-maintained data table, and the API used to store and retrieve the data is based on the file control API used for user-maintained data tables.

### Coupling facility data table structures and servers

Coupling facility data tables are held in coupling facility structures, and access to a coupling facility data table is through a named server. Coupling facility data tables allows you to have related groups of coupling facility data tables stored in separate pools. For example, you might want to have one pool for production and another for test.

Within each MVS image, there must be one CFDT server for each CFDT pool accessed by CICS regions in the MVS image. Coupling facility data table pools are defined as a list structure in the coupling facility resource management (CFRM) policy. The pool name, which is used to form the server name with the prefix DFHCF., is specified in the start-up JCL for the server.

Coupling facility data table pools can be used almost continuously and permanently. CICS provides utility commands that you can use to minimize the impact of maintenance.

Figure 51 on page 384 illustrates a Parallel Sysplex<sup>®</sup> with three CICS AORs linked to the coupling facility data table servers.

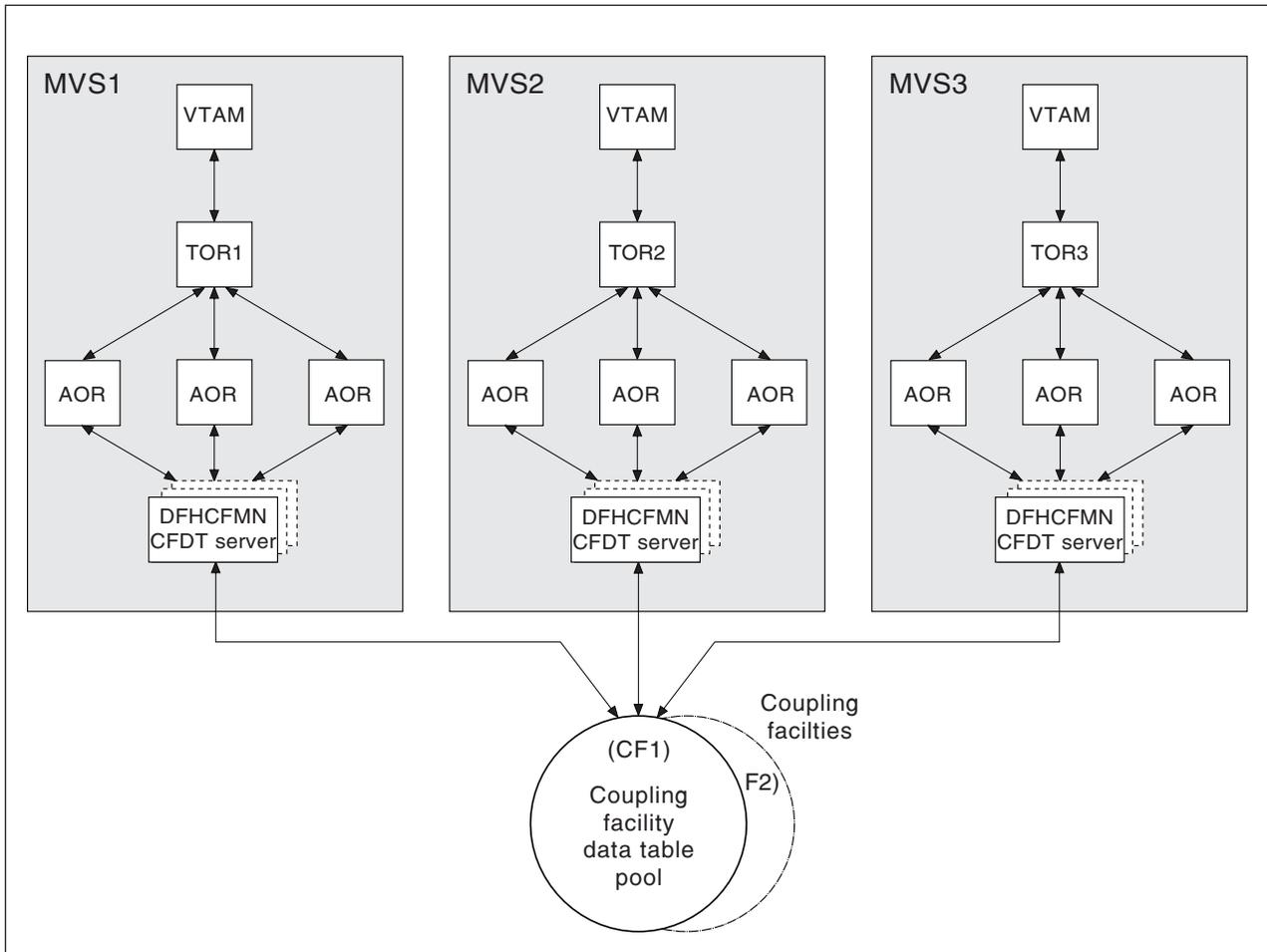


Figure 51. Conceptual view of a Parallel Sysplex with coupling facility data table servers

## Security

The server must be authorized to access the coupling facility list structure in which the data table pool is defined; XES checks this. The server must also be authorized to act as a data table server; AXM checks this. For information on how to define the necessary authorizations see the *CICS RACF Security Guide*.

## Defining and starting a coupling facility data table server region

You activate a coupling facility data table pool in an MVS image by starting up a coupling facility data table server region for that pool. You can start the server as a started task, started job, or as a batch job.

## The server region program, DFHCFMN

The coupling facility data table server region program is called DFHCFMN and must be run from an APF-authorized library. DFHCFMN is supplied in the CICS authorized library, CICSTS31.CICS.SDFHAUTH.

## **SYSIN and SYSPRINT DD statements**

The server reads its initialization parameters from a SYSIN data set, and writes messages and statistics to a print file, for which it requires a SYSPRINT DD statement.

### **Startup parameters**

The coupling facility data table pool server requires some parameters in the start-up JCL. You can specify these in the PARM string, or in the SYSIN data set, or in a combination of both. When a parameter is specified in both places, the PARM value overrides the SYSIN value (because the PARM can be overridden on the MVS START command).

The most important parameter is the pool name, which is mandatory. Among other things, the pool name is used to form, with the prefix DFHCF., the server name (giving DFHCF.*poolname*). Optional pool-related parameters include the maximum number of tables to be supported.

The easiest way to ensure that all pool-related parameters are consistent across MVS images is to use the same SYSIN parameter data set (or an identical copy of it) for all servers accessing the same pool, and to specify in the PARM field any parameters that vary between servers

For details of all the parameters, see “Coupling facility data table server parameters” on page 386.

**Coupling facility data table server REGION parameter:** Use the JCL REGION parameter to ensure that the coupling facility data table server region has enough storage to process the maximum number of data table requests that can be executing concurrently.

The number of coupling facility data table requests that each connected CICS region can have active at a time is limited to about 10. Each request requires about 40KB, therefore the REGION size should specify at least 400KB for each connected CICS region, plus a margin of about 10% for other storage areas. Thus, for a server supporting up to 5 CICS regions, specify REGION=2200K.

During server initialization, the server acquires all the available storage above 16MB, as determined by the REGION parameter, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below 16MB for use in routines that require 24-bit addressable storage, for example sequential file read and write routines.

After initialization, the server uses AXM page allocation services to manage its storage. Server statistics indicate how much storage is actually allocated and used within the storage areas above and below 16MB, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate. Therefore, it is best to ensure that the REGION size is large enough to eliminate this risk.

## Coupling facility data table server JCL example

```
//PRODCFD1 JOB ...
//CFSERVER EXEC PGM=DFHCFMN,REGION=40M,TIME=NOLIMIT CICS CFDT Server
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=* Messages and statistics
//SYSIN DD *
POOLNAME=PRODCFD1 Pool name
MAXTABLES=100 Allow up to 100 tables
/*
```

**Note:** You are recommended to specify `TIME=NOLIMIT`. The server task remains in a wait during most normal processing, because server processing is performed under the client CICS region TCB. If you omit the `TIME` subparameter, your server job could fail with abend S522 (wait limit exceeded), depending on the `JWT` value specified in the `SMFPRMxx` member of `SYS1.PARMLIB`.

Figure 52. Sample JCL to start a CFDT server address space

## Coupling facility data table server parameters

Parameters are specified in the form `KEYWORD=value`, where keywords can optionally be specified in mixed case to improve readability. If you specify more than one parameter in the `PARM` field or on the same `SYSIN` input line, the parameters must be separated by a comma. Any text following one or more spaces is taken as a descriptive comment. Any parameter line which starts with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as in abbreviated or truncated form.

The main parameters are listed on the server print file during start-up.

The parameter descriptions that follow are divided into a number of categories:

- Pool name parameter (on page 386)
- Security parameters (on page 387)
- Statistics parameters (on page 388)
- List structure parameters (on page 389)
- Debug trace parameters (on page 390)
- Tuning parameters (on page 390)
- Lock wait parameters (on page 391)
- Warning parameters (on page 392)
- Automatic structure alter parameters (on page 393)
- Reserved space parameters (on page 393).

The parameters in the above groups are all valid as initialization parameters (in the `SYSIN` file or `PARM` field), and some can also be modified by the `SET` command.

### Pool name parameter

This parameter, `POOLNAME`, is always required:

**POOLNAME=name**

specifies the 8-character name of the coupling facility data table pool. This is appended by the server to the prefix `DFHCF` to create its own server name, as in `DFHCF.poolname`, and also to the prefix `DFHCFLS_` to create the name of the coupling facility list structure, as in `DFHCFLS_poolname`.

This parameter is valid only at server initialization, and must always be specified.

This keyword can be abbreviated to **POOL**.

## Security parameters

You can use these parameters to specify whether you want to use the optional security mechanism that the server provides, to check that CICS regions are authorized to open a coupling facility data table. They also allow you to override standard processing for this optional security.

### **SECURITY={YES|NO}**

specifies whether individual coupling facility data table security checks are required.

**YES** You want the server to perform a security check against each CICS region that attempts to open a coupling facility data table. Access is controlled through profiles defined in the general resource class named on the SECURITYCLASS parameter.

This requires an external security manager, such as RACF, that supports the FASTAUTH function in cross-memory mode.

**NO** You do not want the server to perform this extra security check when opening a coupling facility data table.

This is the only security check performed by the server that is optional. The other file security checks are always performed by the server, as described in the *CICS RACF Security Guide*.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SEC**.

### **SECURITYCLASS={FCICSFCT|class}**

specifies the name of the RACF general resource class that the server is to use for security checks on coupling facility data table access by CICS regions. The name can be up to 8 characters, and is the name of the class in which the CFDT resource profile and its access list are defined.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECCLASS**.

### **SECURITYPREFIX={NO|YES}**

specifies whether the resource name that is passed to RACF for the coupling facility data table security check, when SECURITY=YES is specified, should be prefixed with the server region user ID.

**Note:** For this security check, the resource name used by the server is the either the name specified on the TABLENAME attribute of the CICS file resource definition, or the FILE name if TABLENAME is not specified.

**YES** The server prefixes the resource name with the server region user ID (the default) or an alternative prefix specified on the SECURITYPREFIXID parameter.

**NO** The server passes to RACF only the 8-character resource name, without any prefix.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECPREFIX** or **SECPRFX**.

**SECURITYPREFIXID**=*identifier*

specifies an alternative prefix that the server is to use for security checks on coupling facility data table access by CICS regions, instead of the server region user ID. The prefix can be up to 8 characters, and should correspond to the prefix used to define profile names of CFDTs to RACF. This parameter is effective only if you specify SECURITYPREFIX=YES.

This parameter is valid only at server initialization.

This keyword can be abbreviated to **SECPREFIXID**.

## Statistics parameters

Use the following parameters to specify server statistics options:

**ENDOFDAY**={00:00|*hh:mm*}

specifies the time of day, in hours and minutes, when the server is to collect and reset end-of-day statistics.

**Note:** If the STATSOPTIONS parameter specifies NONE, the server still writes end-of-day statistics to the print file.

The valid range of times is from 00:00 to 24:00.

This keyword can be abbreviated to **EOD**.

**STATSINTERVAL**={03:00|*hh:mm*}

specifies the statistics collection interval, in the range 1 minute to 24 hours. This parameter is ignored if the STATSOPTIONS parameter specifies NONE.

The time interval can range from 00:01 to 24:00.

This keyword can be abbreviated to **STATSINT**.

**STATSOPTIONS**={NONE|**SMF**|**PRINT**|**BOTH**}

specifies whether the server is to produce interval statistics, and the destination for the statistics it produces.

**NONE** The server does not produce any interval statistics.

**SMF** The server produces interval statistics and writes them to the current SMF data set only.

**PRINT** The server produces interval statistics and writes them to the server's print file only.

**BOTH** The server produces interval statistics and writes them to the current SMF data set and to the server's print file.

This keyword can be abbreviated to **STATSOPT**.

## Automatic restart manager (ARM) parameters

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails.

Use the following parameters to override default processing for the automatic restart manager:

**ARMELEMENTNAME**=*elementname*

specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 \$ # @ and the underscore symbol (\_).

The default identifier is of the form DFHCF*nn\_poolname*, where CF represents the server type, *nn* is the &SYSCClone value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

**ARMELEMENTTYPE**=*elementtype*

specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 \$ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

### List structure parameters

These parameters specify list structure attributes. They are used only for the initial allocation of resources when the pool list structure is being created, which occurs the first time you start a server for a pool.

**MAXTABLES**={1000|*number*}

specifies the maximum number of data lists to be reserved when the structure is allocated. The number of data lists determines the maximum number of tables that can be stored in the structure. Note that this parameter also determines how many list headers are defined when the structure is created. Although you should take care to specify a large enough number, specifying an excessively large number will use up an unnecessary amount of coupling facility storage for preallocated list headers.

You cannot change this number without reallocating the structure, which means first deleting the existing structure (see “Deleting or emptying coupling facility data table pools” on page 402). If the structure is being allocated at less than its maximum size, specify a number for the maximum number of tables based on the maximum size of the structure, rather than its initial allocation size.

This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is from 1 to 999 999.

This keyword can be abbreviated to **MAXT**.

**POOLSIZE**={0|*number*}

specifies the initial amount of coupling facility storage to be allocated for the pool list structure, expressed as kilobytes in the form *number*K, megabytes in the form *number*M or gigabytes in the form *number*G.

**0** The special value 0 means that the server is to obtain an initial allocation using the parameters specified in the CFRM policy. If the CFRM policy specifies an INITSIZE value for the structure, this determines the initial allocation, otherwise the maximum SIZE value is allocated.

*number*

A non-zero value specifies an initial amount of storage to be allocated, overriding the INITSIZE parameter in the CFRM policy. This value is generally rounded up by MVS to the next multiple of 256K. The valid range of values is 1K to 2G, but should not be greater than the value specified on the SIZE parameter.

It is generally preferable to omit this parameter, and specify the structure size using the INITSIZE parameter in the CFRM policy. The POOLSIZE option can, however, be useful if the structure is being reallocated or reloaded, and the CFRM policy has not been updated to reflect the required size.

**Note:** If the value is greater than the value specified on the CFRM SIZE parameter, the server POOLSIZE parameter is ignored and the initial allocation is based on the parameters specified in the CFRM policy.

This parameter is valid only at server initialization and is only used when the structure is first allocated.

## Debug trace parameters

These parameters are provided only for intensive debug tracing.

Using these options in a production environment could have a significant impact on performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests can be lost if they are generated faster than the trace print subtask can print them. In this event, the trace only indicates how many messages were lost.

**CFTRACE**={OFF|ON}

specifies the coupling facility interface debug trace option.

**OFF** Coupling facility interface debug trace is disabled.

**ON** Coupling facility interface debug trace produces trace messages on the print file, indicating the main parameters to the coupling facility request interface, and the result from the IXLLIST macro.

This keyword can also be specified as **TRACECF**.

**RQTRACE**={OFF|ON}

specifies the table request debug trace option.

**OFF** Table request debug trace is disabled.

**ON** Table request debug trace produces trace messages on the print file, indicating the main parameters on entry to each cross-memory request, and the results on exit.

This keyword can also be specified as **TRACERQ=**.

## Tuning parameters

These parameters are provided for tuning purposes, but normally you can omit these and let the server assume their default values.

**ELEMENTRATIO**={1|number}

specifies the element part of the entry-to-element ratio when the structure is first allocated. This determines what proportion of the structure space is initially set aside for data elements. (For information about list structures and entry-to-element ratios, see the *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771.)

The valid range is from 1 to 255.

Divide the average size of data per entry by the element size to obtain the optimum value for this ratio. However, if the structure becomes short of space

and altering the ratio could improve space utilization, the server automatically adjusts the ratio according to the actual entry and element usage.

This parameter is valid only at server initialization and is used only when the structure is first allocated.

This keyword can be abbreviated to **ELEMNRATIO**.

**ELEMENTSIZE**={256 | *size*}

specifies the data element size for the list structure, which must be a power of 2. The valid range is from 256 to 4096.

For current coupling facility implementations there is no known reason to use any value other than the default value of 256.

This parameter is valid only at server initialization and is only used when the structure is first allocated.

This keyword can be abbreviated to **ELEMSIZE**.

**ENTRYRATIO**={1 | *number*}

specifies the entry part of the entry-to-element ratio when the structure is first allocated. This determines what proportion of the structure space is initially set aside for list entry controls. (For information about list structures and entry-to-element ratios, see the *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771.)

It is not essential to specify this parameter because the server automatically adjusts the ratio based on actual usage to improve space utilization if necessary.

This parameter is valid only at server initialization and is used only when the structure is first allocated. The valid range is from 1 to 255.

## Lock wait parameters

Use these parameters to modify the time intervals for the server lock wait retry mechanism. This is provided mainly as a “wake-up” mechanism to ensure that requests waiting for a record lock do not wait indefinitely in certain rare cases where a system failure or structure full condition could cause a lock release notification message to be lost. In addition, this mechanism also ensures that if a CICS task is purged while it has a request waiting for a lock, the waiting request in the server is woken up as soon as the lock wait retry interval expires. The request process then finds that the CICS task is no longer waiting for it, therefore it terminates rather than continuing to wait and hold on to server resources until the lock becomes free.

There are two times involved: a scan time interval and a wait time. The server starts its lock scan interval timing when the first request is made to wait.

This mechanism has very little effect on normal processing and the default lock wait retry parameter values are designed to suit the majority of installations.

**LOCKSCANINTERVAL**={5 | *number*}

specifies the time interval after which requests waiting for record locks are scanned to check for lock wait timeout.

This affects the overall duration of the lock wait timeout, because a request that starts waiting for a lock during a given scan interval is timed as if from the start of the interval. The lock scan interval should be less than the lock wait interval, and ideally should be such that the lock wait interval is an exact multiple of the lock scan interval.

You can specify this value as a number of seconds or in the time format *hh:mm:ss*.

The valid range is from 1 (1 second) to 24:00 (24 hours).

This keyword can be abbreviated to **LOCKSCANINT**.

**LOCKWAITINTERVAL**={10|*number*}

specifies the maximum time that a request should wait for a record lock before being reinvoked to retry. The actual time a request waits depends on how far into a scan interval it started its wait. For example, in a server using the scan and wait default intervals, if 4 seconds have already elapsed when a request starts to wait, the maximum time it can wait is 6 seconds. When the server checks the timeout value for the request, it assumes that the request has been waiting for the full scan period. Thus, for the default values, a request is reinvoked to retry after waiting between five and ten seconds.

This value can either be specified as a number of seconds or in time interval format *hh:mm:ss*.

The valid range is from 1 (1 second) to 24:00 (24 hours).

This keyword can be abbreviated to **LOCKWAITINT**.

### Warning parameters

Use these parameters to modify the thresholds at which warning messages are issued, and an automatic structure alter occurs, when the structure becomes nearly full.

**ELEMENTWARN**={80|*number*}

specifies the percentage of list structure elements in use at which warning messages and an automatic structure alter should be first triggered.

The valid range is from 1 to 100 per cent.

This keyword can be abbreviated to **ELEMWARN**.

**ELEMENTWARNINC**={5|*number*}

specifies the percentage increase (or decrease) of elements in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of elements in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

This keyword can be abbreviated to **ELEMWARNINC**.

**ENTRYWARN**={80|*number*}

specifies the percentage of list structure entries in use at which warning messages and an automatic structure alter action should be first triggered.

The valid range is from 1 to 100 per cent.

**ENTRYWARNINC**={5|*number*}

specifies the percentage increase (or decrease) of entries in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of entries in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

## Automatic structure alter parameters

Use these parameters to modify the conditions under which the server attempts an automatic alter when the structure becomes nearly full. For information about the structure alter process, see “Coupling facility data table server automatic structure alter” on page 395.

**ALTERELEMMIN**={100|*number*}

specifies the minimum number of excess elements that must be present to cause the server to issue an automatic alter to convert those elements to entries.

The valid range is from 0 to 999999999.

**ALTERELEMPC**={5|*number*}

specifies the minimum percentage of excess elements that must be present to cause the server to issue an automatic alter to increase the proportion of entries.

The valid range is from 0 to 100 per cent.

**ALTERENTRYMIN**={100|*number*}

specifies the minimum number of excess entries that must be present to cause the server to issue an automatic alter to convert those entries to elements.

The valid range is from 0 to 999999999.

**ALTERENTRYPC**={5|*number*}

specifies the minimum percentage of excess entries that must be present to cause the server to issue an automatic alter to increase the proportion of elements.

The valid range is from 0 to 100 per cent.

**ALTERMININTERVAL**={00:10|*hh:mm*}

specifies the minimum time interval to be left between automatic structure alter attempts when the structure is nearly full (above the element or entry warning level). This is to prevent excessive numbers of alter attempts to try to optimize space when the structure is nearly full.

The valid range is from 00:00 to 24:00.

This keyword can be abbreviated to **ALTERMININT**.

## Reserved space parameters

Use these parameters to control the amount of structure space that is reserved for rewrites and server internal operations (such as tracking units of work and notifying other servers when a lock is released). For information about the effect of these parameters, see “Avoiding structure full conditions” on page 394.

**ELEMENTRESERVEMIN**={300|*number*}

specifies the minimum number of list structure data elements (normally 256 bytes each) to be reserved for rewrites and server internal operations.

The valid range is from 0 to 999999999.

This keyword can be abbreviated to **ELEMRESERVEMIN**.

**ELEMENTRESERVEPC**={5|*number*}

specifies the percentage of list structure data elements to be reserved for rewrites and internal server use. If this percentage evaluates to less than the minimum number specified on the **ELEMENTRESERVEMIN** parameter, the minimum number is used instead.

The valid range is from 0 to 100.

This keyword can be abbreviated to **ELEMRESERVEPC** or **ELEMRESPC**.

**ENTRYRESERVEMIN**={100|*number*}

specifies the minimum number of list structure entries to be reserved for rewrites and server internal operations.

The valid range is from 0 to 999999999.

This keyword can be abbreviated to **ENTRYRESMIN**.

**ENTRYRESERVEPC**={5|*number*}

specifies the percentage of list structure elements to be reserved for rewrites and internal server use. If this percentage evaluates to less than the minimum number specified on the ENTRYRESERVEMIN parameter, the minimum number is used instead.

The valid range is from 0 to 100.

This keyword can be abbreviated to **ENTRYRESPC**.

## Avoiding structure full conditions

If the coupling facility data table structure is allowed to become completely full, this not only prevents the addition of new records or tables, but can also have a significant impact on performance and application function. In particular, rewrite requests can be rejected even when the size of the new data is less than or equal to the original, and server internal operations can fail, causing internal time-outs and retries.

The parameters ELEMENTRESERVEMIN, ELEMENTRESERVEPC, ENTRYRESERVEMIN and ENTRYRESERVEPC are provided to reduce the risk of the structure becoming totally full, by reserving a number of entries and elements, which can only be used for operations that normally only need extra space temporarily, such as rewrites or unit of work control operations. If a server is asked to write a new record or create a new table when the number of entries or elements remaining in the structure (as returned by each coupling facility access request) is less than or equal to the specified reserve level, the request is rejected with an indication that no space is available. Before rejecting the request, the server issues a dummy read request in order to find out the latest usage levels for the structure, in case more space has recently become available.

Using the reserved space parameters means that, even if the structure fills up very rapidly (for example, because a table is being loaded that is too large for the available space), enough space should remain to allow rewrites of existing records and allow internal communication between servers to continue normally.

Note that this mechanism cannot prevent the structure from eventually becoming totally full, as recoverable rewrites are allowed to use the reserved space temporarily, and rewrites that increase the data length will gradually use up the reserved elements. If action is not taken to prevent the structure from becoming totally full, the following effects can occur:

- An attempt to close a table or change the table status could encounter a temporary structure full condition. In this case, the attempt is retried indefinitely, because it must be completed in order to preserve table integrity (the only alternative being to terminate the server). The retry process normally succeeds quickly, but there is a theoretical case where this can cause a loop until another server causes temporarily unavailable resources to be released.
- Rewrites with the same (or smaller) data size for a table using the contention update model are retried indefinitely if they initially fail because of a structure full

condition. This is done to protect the application against having to handle this unexpected form of failure. Again, the retry should normally succeed quickly, but there is a theoretical possibility that this could loop for a while.

- Rewrites for a table using the locking or recoverable update model could be rejected with a structure full condition even if the data size is not increased. No retry is attempted in this case.
- Units of work can be backed out because the server is unable to create unit of work control entries for commit processing.
- There may not be sufficient structure space to send lock release messages, in which case waiting tasks are not woken up immediately but continue to wait for up to the time-out interval specified on the LOCKWAITINTERVAL parameter before finding out that the lock has been released.

## **Coupling facility data table server automatic structure alter**

The coupling facility data table server monitors the total number of elements and entries in use in the structure, using information returned by the coupling facility on every request. When the numbers in use exceed the specified warning thresholds, the server issues a warning message, and this warning message is repeated each time the number in use increases beyond further thresholds.

Each time the server issues a warning, it also tests whether an automatic structure alter for the entry-to-element ratio should be issued. If any form of alter has already been issued recently (by any server or through an operator SETXCF ALTER command) and the structure space usage has remained above warning levels since the previous attempt, any further structure alter attempt is suppressed until at least the minimum interval (specified through the ALTERMININTERVAL parameter) has elapsed.

The server checks whether an automatic structure alter should be issued, by calculating how many excess elements or entries will be left when the other runs out completely, based on the ratio between the current numbers of elements and entries actually in use. If the number of excess elements or entries exceeds the number specified in the ALTERELEMMIN or ALTERENTRYMIN parameter, and the same number expressed as a percentage of the total exceeds the value specified in the ALTERELEMPC or ALTERENTRYPC parameter, an IXLALTER request is issued to alter the entry to element ratio to the actual current ratio between the number of entries and elements in use.

Only one alter request can be active at a time for a given structure. This means a server may well find that another server has already started the structure alter process, in which case its own alter is rejected. However, the system automatically notifies all servers when the structure alter is completed, giving the new numbers of elements and entries so that each server can update its own status information.

---

## Controlling coupling facility data table server regions

You can issue commands to control a coupling facility data table server, using the MVS MODIFY (F) command to specify the job or started task name of the server region, followed by the server command. The general form of a coupling facility data table server command, using the short form F, is as follows:

```
F job_name,command parameters... comments
```

The commands supported by a coupling facility data table server are:

- SET
- DISPLAY
- PRINT
- DELETE TABLE
- STOP
- CANCEL

These commands and their options are as follows:

**SET** *keyword=operand[,keyword=operand,...]*

Change one or more server parameter values. The command can be abbreviated to **T**, as for the MVS SET command. See “The SET command options” on page 397 for details.

**DISPLAY** *keyword[=operand][,keyword[=operand],...]*

Display one or more parameter values, or statistics summary information, on the console. The valid keywords for DISPLAY are all the initialization parameters, plus an additional set described under “DISPLAY and PRINT command options” on page 398.

The command can be abbreviated to **D**, as for the MVS DISPLAY command.

**PRINT** *keyword[=operand][,keyword[=operand],...]*

Produces the same output as DISPLAY, supporting the same keywords, but on the print file only.

**DELETE TABLE**=*name*

Delete the named table. The table must not be in use for this command to succeed. The command can be abbreviated to **DEL**.

**STOP**

Terminate the server normally. The server waits for any active connections to terminate first, and prevents any new connections while it is waiting. The command can be abbreviated to **P**, as for the MVS STOP command.

**Note:** You can also use the MVS STOP command.

```
P jobname
```

This is equivalent to issuing the server STOP command through the MVS MODIFY command.

**CANCEL** {RESTART={**NO**|**YES**}}

Terminate the server immediately. You can specify whether automatic restart should be requested.

For information about CANCEL RESTART see “The CANCEL command options” on page 401.

## The SET command options

You can use the SET command to modify the following groups of server initialization parameters:

- The statistics parameters
- The debug trace parameters
- The lock wait parameters
- The warning parameters
- The automatic ALTER parameters.

See “Coupling facility data table server parameters” on page 386 for details of these keywords.

The following **SET** keywords are used to modify the server's recovery status of an inactive CICS region that had unresolved units of work when it last terminated:

**RESTARTED**=*applid*

Establish a temporary recoverable connection for the given APPLID. This resolves any units of work that were in commit or backout processing when the region last terminated, and indicates whether there are any remaining in-doubt units of work.

This keyword can be abbreviated to **RESTART** or **REST**.

**COMMITTED**={*applid*|*applid.uowid*}

Establish a temporary recoverable connection for the specified APPLID and commit all in-doubt units of work, or, if *uowid* is also specified, commit that specific unit of work.

This command should be used **only** when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **COMMIT** or **COMM**.

**BACKEDOUT**={*applid*|*applid.uowid*}

Establish a temporary recoverable connection for the specified APPLID and back out all in-doubt units of work, or, if *uowid* is also specified, back out that specific unit of work.

This command should be used *only* when it is not possible to restart the original CICS region to resolve the work normally, because it can result in inconsistency between coupling facility data table resources and other CICS resources updated by the same unit of work.

This keyword can be abbreviated to **BACKOUT** or **BACK**.

Use the following **SET** parameters to modify options relating to a specific table:

**TABLE**=*name*

specifies the table to which the following table-related parameters in the same command are to be applied. This parameter is required before any table-related parameters.

**MAXRECS**=*number*

Modify the maximum number of records that can be stored in the table specified by the preceding **TABLE** parameter.

If the maximum number is set to a value less than the current number of records in the table, no new records can be stored until records have been deleted to reduce the current number to within the new maximum limit. For a

recoverable table, this also means that records cannot be updated, because the recoverable update process adds a new record on the rewrite operation then deletes the original record when the transaction completes.

This keyword can also be specified as **MAXNUMRECS**.

**AVAILABLE={YES|NO}**

Specify whether the table named by the preceding **TABLE** parameter is available for new OPEN requests. If the table is made unavailable, a CICS region that subsequently issues an OPEN request for the table receives a response indicating that it is unavailable, but regions that currently have the table open are not affected. Even when a table is marked as unavailable, a server can implicitly open it on behalf of a CICS region to allow recoverable work to be resolved during restart processing.

This keyword can be abbreviated to **AVAIL**.

**Examples of the SET command:** The following example changes the statistics options:

```
SET STATSOPT=BOTH,EOD=21:00,STATSINT=06:00
```

The following example modifies the maximum number of records allowed in the specified table:

```
SET TABLE=PAYECFT1,MAXRECS=200000
```

## DISPLAY and PRINT command options

You can use the DISPLAY (and PRINT) commands to display the values of any initialization parameters plus some additional information.

Some of the parameters that provide additional information support generic names. You specify generic names using the following wildcard characters:

- An \* (asterisk symbol). Use this anywhere in the parameter value to represent from 0 to 8 characters of any value. For example, CICSH\* to represent all the CICS APPLIDs in a CICSplex identified by the letter H.
- A % (per cent symbol). Use this anywhere in the parameter value to represent only one character of any value. For example, CICS%T\* to represent all the TOR APPLIDs in all CICSplexes.

The parameters supported by the DISPLAY and PRINT commands are as follows:

**APPLIDS**

Display the APPLID and MVS system name for every CICS region that currently has a recoverable connection to the pool. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

This keyword can be abbreviated to **APPLID**, **APPLS** or **APPL**.

**APPLID={*applid*|*generic*}**

Display the APPLID and MVS system name for each region that currently has a recoverable connection to the server's pool, and whose APPLID matches *applid* or *generic*. This command returns information not only for the server to which the MODIFY command is issued, but for all other servers connected to the same pool.

*applid* Use this for a specific APPLID, which should match only one region in the sysplex.

### *generic*

Use a suitable generic value when you want to obtain information about several regions.

If *applid* or *generic* is not specified, the server treats this as equivalent to the command DISPLAY APPLIDS.

This keyword can also be specified as **APPLIDS**, **APPLS** or **APPL**.

### **ARMREGISTERED**

Shows whether ARM registration was successful (YES or NO).

### **CONNECTIONS**

Display the jobnames and applids of the regions currently connected to the server to which the command is issued.

This keyword can be abbreviated to **CONN**.

### **TABLES**

Display the names of all tables currently allocated in the pool.

**TABLE**={*name*|*generic\_name*}

Display information about the attributes and status of a specific table, or of a set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLES.

### **TABLEUSERS**

Display the CICS APPLIDs of the regions that are currently using each of the tables currently defined in the pool.

This keyword can be abbreviated to **TABLEU**.

**TABLEUSERS**={*name*|*generic\_name*}

Display the CICS APPLIDs of the regions that are currently using the specified table, or using each of the set of tables whose names match the generic name.

If no table name is specified, this is treated as equivalent to DISPLAY TABLEUSERS.

This keyword can be abbreviated to **TABLEU**

### **UOWIDS**

Display the applids of all regions that currently have unresolved recoverable units of work, together with the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

This keyword can be abbreviated to **UOWS**.

**UOWIDS**={*applid*|*generic\_applid*}|{*applid.\**|*generic\_applid.\**}

Display, for the specified regions if they currently have unresolved recoverable units of work, information about those units of work. The information returned depends on the form of operand used.

*applid*|*generic\_applid*

This form of operand displays simply the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid*, the server displays UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic\_applid* the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

*applid.\*|generic\_applid.\**

This form of operand displays:

- The state and local UOWID of each individual unit of work, followed by
- A summary of the number of units of work that are currently in doubt, or are in the process of being committed or backed out.

If you specify *applid.\**, the server displays the UOW information for a specific APPLID, which should correspond to only one region in the sysplex.

If you specify *generic\_applid.\**, the server displays UOW information for all the APPLIDs that match the generic APPLID specified.

This keyword can be abbreviated to **UOWS**.

**UOWID**=*applid.uowid*

Display the state of an individual unresolved unit of work, identified by its applid and local unit of work ID (UOWID). Enter the local UOWID as 16 hexadecimal digits.

This keyword can be abbreviated to **UOW**.

## **DISPLAY and PRINT options for statistics summaries**

Use the following parameters to display or print statistics:

### **CFSTATS**

Display statistics for coupling facility interface accesses and responses from the server.

This keyword can also be specified as **CFST** or **STATSCF**.

### **POOLSTATS**

Display usage statistics for the pool list structure as a whole. This is based on information returned by coupling facility access requests, therefore it is only as current as the most recent request made through the server to which the command is issued.

This keyword can be abbreviated to **POOLST**.

### **TABLESTATS**

Display statistics for requests, processed by the server to which the command is issued, for each table plus a summary of all requests processed, including those that are not table-specific, such as unit of work control.

Note that only tables with a non-zero number of requests since the start of the current statistics interval are shown.

This keyword can also be specified as **TABLEST**.

**TABLESTATS**={*name*|*generic\_name*}

Display request statistics for the specified table or tables.

*name* A specific table name in the pool accessed by the server. Returns statistics for this table only.

*generic\_name*

A generic name that you can use to obtain statistics about a number of tables. Returns statistics for any table name that matches the generic name.

This keyword can be abbreviated to **TABLEST**.

### **STORAGESTATS**

Display main storage allocation statistics for the server address space.

This keyword can be abbreviated to **STORAGEST** or **STGST**.

### **DISPLAY and PRINT options for combined lists of information**

These keywords represent combined lists of information:

#### **PARAMETERS**

Display the main parameter values. These are POOLNAME, SECURITY, SECURITYPREFIX, statistics options, and list structure options.

This keyword can be abbreviated to **PARM** or **PARMS**.

#### **ALLPARAMETERS**

Display all parameter values.

This keyword can be abbreviated to **ALLPARMS**.

#### **STATISTICS**

Display all available statistics.

This keyword can be abbreviated to **STAT** or **STATS**.

#### **INITIALIZED**

Display the parameters and statistics that are usually displayed when initialization is complete. This is equivalent to PARM, POOLSTATS, STGSTATS.

This keyword can be abbreviated to **INIT**.

#### **ARM**

Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

## **The CANCEL command options**

You can use the CANCEL command to request automatic restart by specifying the following parameter:

#### **RESTART={NO|YES}**

Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname, ARMRESTART.

You can also enter RESTART on its own for RESTART=YES, NORESTART for RESTART=NO.

## SETXCF commands

The server also responds to XES events such as an operator **SETXCF** command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server **CANCEL** command to close itself down immediately.

---

### Deleting or emptying coupling facility data table pools

You can delete a coupling facility data table pool using the MVS **SETXCF** command to delete its coupling facility list structure:

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHCFLS_poolname
```

You can delete a structure only when there are no servers connected to the pool, otherwise MVS rejects the command.

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy, and other values determined by the server initialization parameters (in particular, **MAXTABLES**).

---

### Unloading and reloading coupling facility data table pools

You can unload, and reload, the complete contents of a coupling facility data table pool to and from a sequential data set by invoking the server program with the **FUNCTION** parameter, using the **UNLOAD** and **RELOAD** options. The unload and reload process preserves not only the table data, but also all recovery information such as unit of work status and record locks for recoverable updates.

You can use this function, for example, to

- Preserve the coupling facility data table pool during planned coupling facility maintenance, or
- Move the pool to a different coupling facility.
- Increase the size of the pool's list structure.

If the maximum number of tables specified in the original pool was too small, or the pool has reached its maximum size and needs to be expanded further, unload the pool, then delete the structure so that the reload process can reallocate it with more space.

**FUNCTION={UNLOAD|RELOAD}**

Specify the function for which the server is being initialized.

#### **UNLOAD**

Unload the entire contents of the coupling facility data table pool specified on the **POOLNAME** parameter to a sequential data set. When the unload processing has completed (normally or abnormally) the server program terminates.

The **UNLOAD** function requires a DD statement for **DDNAME DFHCFUL** describing the sequential data set to which the table pool is to be unloaded. The format of the unloaded data set is:

```
RECFM=F  
LRECL=4096  
BLKSIZE=4096
```

You can obtain an estimate of the upper limit for the total size of the data set, in bytes, from the pool usage statistics produced by the server:

- From the statistics, multiply the number of elements in use by the element size (usually 256) to get a total number of bytes for the data size, although the space actually needed to unload the data is normally much less, because unused space in a data element is not unloaded.
- Add some space for the record keys, calculated using a two-byte prefix plus the keylength for each record, plus about 100 bytes per table for table control information. Thus, the maximum you should need for keys and control information is:

$(18 \text{ bytes} \times \text{number of entries}) + (100 \text{ bytes} \times \text{number of tables})$

## RELOAD

Reload, into the coupling facility data table pool named on the POOLNAME parameter, a previously unloaded coupling facility data table pool.

You can reload a pool into a pool with a different name—it does not have to keep the same name as the original pool. When the reload processing has completed (normally or abnormally) the server program terminates.

The RELOAD function requires a DD statement for DDNAME DFHCFRL, describing the sequential data set from which the table pool is to be reloaded.

The structure is allocated, if necessary, during reloading, in which case you can use the same server parameters to control structure attributes as for normal server startup. The reload process bypasses any tables or units of work that are already found in the pool (for example, because the structure was too small and the reload job had to be restarted after using ALTER to increase the structure size).

**Note:** If the unloaded pool structure was altered dynamically at any time after initial allocation (by using the SETXCF command to increase the size), ensure that the increased size is allocated for the reloaded pool. The recommended way is to update the INITSIZE parameter for the structure in the current CFRM policy whenever you alter the structure size, and to activate the updated policy using the SETXCF START ,POLICY command. Alternatively, you can specify the required pool size in the POOLSIZE parameter in the reload JCL, but note that this does not override the CFRM INITSIZE parameter if it is exactly equal to the maximum pool size.

**Note:** If you omit the FUNCTION parameter, the server program initializes a coupling facility data table server address space.

For the UNLOAD and RELOAD function, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the unload or reload attempt is rejected. Similarly, if a normal server attempts to start up while an unload or reload job is in progress, the attempt fails because shared access to the structure is not available.

You can specify all normal server parameters when unloading or reloading, but some of these (for example, security-related parameters) are ignored because they do not apply to unload or reload processing.

Note that when a pool is nearly full (with less than about 5% free entries and elements) there is no guarantee that it can be unloaded and reloaded into a structure of exactly the same size. This is because the amount of space available is affected by the current ratio of entries to elements, which is controlled only approximately by the automatic ALTER process. If the structure reaches the warning level during reloading, the automatic ALTER process attempts to adjust the entry to element ratio. The reload process automatically waits for the ALTER to complete if reloading runs out of space while an ALTER is still in progress.

If reloading fails because it runs out of space, the resulting messages include the numbers of tables reloaded and blocks read up to the time of the failure. You can compare these values with those in the messages from the original unload job, to determine how many more tables and how much more data remains to be loaded. If a table had been partially reloaded before running out of space, it is deleted so that the whole table is reloaded again if the reload is retried later. If reloading is interrupted for any other reason than running out of space, for example by an MVS system failure, reloading can still be restarted using the partially reloaded structure, but in that case the structure space occupied by any partially reloaded table will be unavailable, so it is normally better to delete the structure (using the MVS **SETXCF FORCE** command) and start reloading again with a newly allocated structure.

```
//UNLDCFD1 JOB ...
//DTUNLOAD EXEC PGM=DFHCFMN          CICS CF data table server program
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=*                Options, messages and statistics
//DFHCFUL DD DSN=CFDT1.UNLOADED.POOL, Unloaded data table pool
//      DISP=(NEW,CATLG),
//      SPACE=(4096,(10000,1000)) Estimated size in 4K blocks
//SYSIN DD *
FUNCTION=UNLOAD                      Function to be performed is UNLOAD
POOLNAME=PRODCFD1                   Pool name
/*
```

Figure 53. Unload JCL example

```
//RELDCFD1 JOB ...
//DTRELOAD EXEC PGM=DFHCFMN          CICS CF data table server program
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=*                Options, messages and statistics
//DFHCFRL DD DSN=CFDT1.UNLOADED.POOL,DISP=OLD Unloaded table pool
//SYSIN DD *
FUNCTION=RELOAD                      Function to be performed is RELOAD
POOLNAME=PRODCFD1                   Pool name
POOLSIZE=50M                        Increased pool size
MAXTABLES=500                       Increased max number of tables
/*
```

Figure 54. Reload JCL example

---

## Chapter 26. Setting up and running a named counter server

This chapter describes how to set up and run a named counter server, and provides information about the following:

- “Named counter server overview”
- “Defining a named counter options table” on page 407
- “Defining a list structure” on page 409
- “Defining and starting a named counter server region” on page 411
- “Controlling named counter server regions” on page 415
- “Changing the size of named counter pools” on page 418
- “Deleting or emptying named counter pools” on page 418
- “Unloading and reloading named counter pools” on page 419
- “Dumping named counter pool list structures” on page 421

---

### Named counter server overview

CICS provides an efficient way of generating unique sequence numbers for use by applications in a Parallel Sysplex environment (for example, to allocate a unique number for orders or invoices). A named counter server maintains each sequence of numbers as a named counter. Each time a number is assigned, the corresponding named counter is automatically incremented so that the next request gets the next number in sequence. Named counters are the Sysplex equivalent of COUNTER in the Common System Area (CSA) of a single region CICS system.

CICS provides a command level API for the named counter facility. See the *Application Programming Guide* for general information, and the *Application Programming Reference* for command syntax. A CALL interface makes the named counters available to batch applications; see the *Application Programming Guide* for details.

A named counter server provides a full set of functions to define and use named counters. Each named counter consists of:

- A 16-byte name
- A current value
- A minimum value
- A maximum value.

The values are internally stored as 8-byte (double word) binary numbers, but the user interface allows them to be treated as any length from 1 to 8 bytes, typically 4 bytes.

Named counters are stored in a pool of named counters, where each pool is a small coupling facility list structure, with keys but no data. The pool name forms part of the list structure name. Each named counter is stored as a list structure entry keyed on the specified name, and each request for the next value requires only a single coupling facility access.

**Warning:** The counters are lost if the coupling facility fails. See the *Application Programming Guide* for details of recovery techniques.

For information on how to create a list structure for use as a named counter pool, see “Defining a list structure” on page 409.

## Named counter structures and servers

Within each MVS image, there must be one named counter server for each named counter pool accessed by CICS regions (and batch jobs) in the MVS image. Named counter pools are defined as a list structure in the coupling facility resource management (CFRM) policy. The pool name, which is used to form the server name with the prefix DFHNC, is specified in the start-up JCL for the server.

Figure 55 illustrates a Parallel Sysplex with three CICS AORs linked to named counter servers.

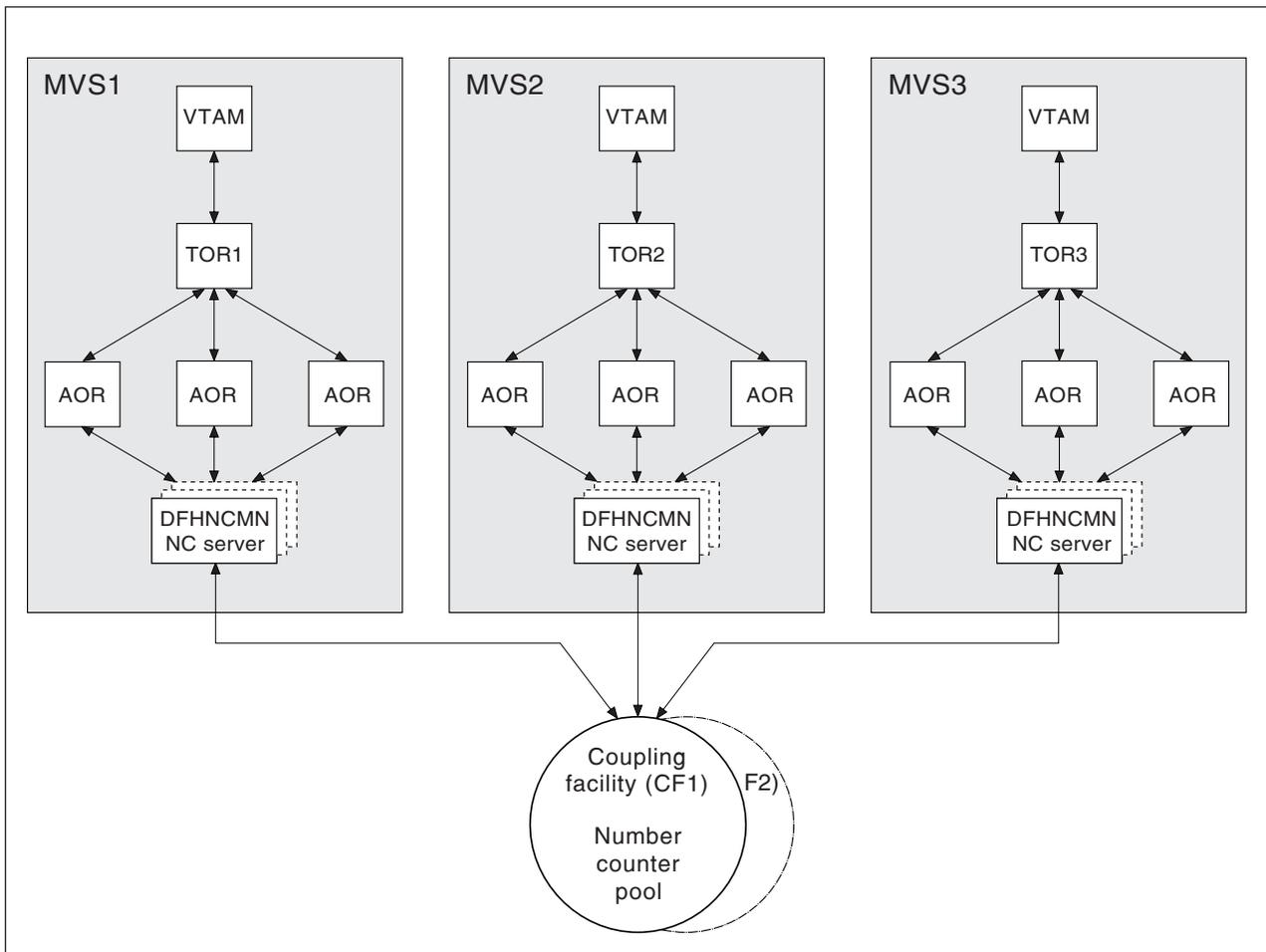


Figure 55. Conceptual view of a Parallel Sysplex with named counter servers

## Selecting a named counter server

To reference a named counter, an application program can specify either the actual name of the pool in which the named counter is stored, or it can specify a dummy pool selection parameter, which is mapped to the actual pool name by the POOL parameter specified in the options table, DFHNCOPT. This makes it easy to use a different pool (for example, to isolate test pools from production pools) without having to change the pool selection parameter in the application program. To vary the pool used by a CICS region, either load a different copy of the options table from STEPLIB, or use a common options table where the pool name selection is conditional on the job name and CICS APPLID, in addition to the pool name selection parameter. The options table also supports invocation of a user-specified program to select the appropriate pool given the pool selection parameter.

## Security

The server must be authorized to access the coupling facility list structure in which the named counter pool is defined; XES checks this. The server must also be authorized to act as a named counter server; AXM checks this. For information on how to define the necessary authorizations see the *CICS RACF Security Guide*.

**Note:** You cannot control access to individual named counters.

---

## Defining a named counter options table

The named counter callable interface determines the actual pool name in response to a DFHNCTR call by referring to the DFHNCOPT options table. When a pool selector value is encountered for the first time, the pool name is determined via the options table. The name is then saved and used for all subsequent requests for the same pool selector from the same TCB. This continues for the life of the TCB or until the NC\_FINISH function is used specifying that pool selector value. CICS supplies a default DFHNCOPT in source form, which you can customize and generate using the DFHNCO macro. A typical use of the options table is to enable production and test regions to use a different counter pool without needing to change the pool name in application programs.

To avoid the need to maintain multiple versions of the options table, you can use table entries to select pools based not only on the pool selection parameter specified on the DFHNCTR call, but also on the job name and APPLID of the CICS region. You can also specify the name of a user exit program to be called to make the pool selection.

Define an options table using one or more invocations of the DFHNCO macro. Each invocation generates an options table entry that defines the pool name or user exit program to be used whenever any selection conditions specified on the entry satisfy an application program request. The first entry automatically generates the table header, including the CSECT statement. Follow the last entry with an END statement specifying the table module entry point, DFHNCOPT.

## The options table parameters

The DFHNCOPT options table parameters are illustrated in Figure 56.

```
DFHNCO [POOLSEL={{(generic_values) |*},}]
        [JOBNAME={{(generic_values) |*},}]
        [APPLID={{(generic_values) |*},}]
        {POOL={YES|NO|name} | CALL=programname}

        Terminate the last DFHNCO entry with the
        following END statement:

END      DFHNCOPT
```

Figure 56. DFHNCOPT options table

The POOLSEL, JOBNAME, and APPLID parameters specify optional selection conditions to determine whether the entry applies to the current request. You can specify each of these operands as

- A single generic name
- A list of names in parentheses, the list containing two or more generic names, each name separated by a comma.

Each name comprises the characters that can appear on the appropriate parameter, plus the wild-card characters \* to match any sequence of zero or more non-blank characters, and % to match any single non-blank character. When multiple generic name are specified, the selection condition is satisfied if any one of them matches. A blank pool selector value can be matched using a null POOLSEL operand, for example POOLSEL= or POOLSEL=().

**POOLSEL={ (generic1, generic2, ..., ...) | \* }**

Specifies that this options table entry applies only when the pool selection parameter specified by the application program matches one of the generic names specified on this parameter.

Specifying POOLSEL=, or POOLSEL=() is the equivalent of specifying 8 blanks.

If you omit the POOLSEL keyword, it defaults to \*.

**JOBNAME={ (generic1, generic2, ..., ...) | \* }**

Specifies that this options table entry applies only when the caller's job name matches one of the generic names specified on this parameter.

If you omit the JOBNAME keyword, it defaults to \*.

**APPLID={ (generic1, generic2, ..., ...) | \* }**

Specifies that this options table entry applies only when the caller's CICS APPLID matches one of the generic names specified on this parameter.

If you omit the APPLID keyword, it defaults to \*.

**POOL={ YES | NO | *name* }**

Specifies the pool name to be used. This parameter is mutually exclusive with the CALL parameter. The options are:

**YES** specifies that the server is to use the pool selection parameter specified by the application program as the actual pool name. An all-blank pool selection parameter means the server is to use the default pool name. For the call interface, the default name is DFHNC001. For the EXEC CICS API, the default name is specified by the NCPLDFT system initialization parameter.

**NO** specifies that the server is not to use any pool and is to reject the request with an error.

*name* specifies the actual pool name that the server is to use. If *name* is omitted, this indicates that the default pool is to be used. (For the CALL interface, the default pool is always DFHNC001, but for the EXEC CICS interface you can specify the default pool using the NCPLDFT system initialization parameter.)

**CALL=*programname***

specifies the name of a user exit program to be called to determine the actual pool name to be used. This parameter is mutually exclusive with the POOL parameter.

The program named can be link-edited with the options table, which generates a weak external reference (WXTRN), or it can be loaded dynamically the first time it is used. The program is called using standard MVS linkage in AMODE 31, with a standard save area and parameter list pointing to four fields, in the following order:

- The 8-byte actual pool name result field
- The 8-byte pool selection parameter.
- The 8-byte job name

- The 8-byte APPLID if running under CICS, otherwise blanks

The end-of-list bit is set in the last parameter address.

The program should be reentrant and should be linked with RMODE ANY, so that it (and the option table if linked with the program) can be loaded above the line. Temporary working storage can be acquired and released using MVS GETMAIN and FREEMAIN. As this program is only called when a new pool selection value is used, the use of GETMAIN and FREEMAIN should not affect performance.

The exit program cannot use any CICS services. If it is used in a CICS region, it must avoid using any MVS services which could result in a long wait, as it will normally be executed under the CICS quasi-reentrant (QR) TCB.

The user exit program indicates its result by setting one of the following return codes in register 15:

- 0** Use the pool name that is successfully set, in the first field of the parameter list, by the user exit program.
- 4** The program cannot determine the pool name on this invocation. Continue options table processing at the next entry, as for the case where selection conditions were not met.
- 8** Reject the request (as if POOL=NO was specified).

The default options table, supplied in CICSTS31.CICS.SDFHLINK, contains the following entries:

```
DFHNCO POOLSEL=DFHNC*,POOL=YES
DFHNCO POOL=
END DFHNCOPT
```

With the default options table in use, any pool selector parameter that specifies a string beginning with DFHNC is taken to be an actual pool name, indicated by POOL=YES in the table entry. Any other value, including a value of all spaces, is assigned the default pool name, indicated by the POOL= table entry without a POOLSEL parameter.

The source for this default table is supplied in CICSTS31.CICS.SDFHSAMP.

## Making an options table available to CICS

To ensure that your CICS region can load the named counter options table, install the link-edited table into a CICS authorized library in STEPLIB. Alternatively you can install the table in a suitable library in the LINK list.

---

## Defining a list structure

Define one or more coupling facility list structures for the named counter facility, each list structure representing a pool of named counters. Each named counter pool is accessed through a cross-memory server region.

Define the structure in the current coupling facility resource management (CFRM) policy, specifying the size of the structure and the preference list of coupling facilities in which it can be stored. The name of the list structure for a named counter pool is formed by adding the prefix DFHNCLS\_ to your chosen pool name, giving DFHNCLS\_*poolname*.

The CFRM policy is defined using the utility IXCMIAPU. For an example of this utility, see member IXCCFRMP in the SYS1.SAMPLIB library. An example of a policy statement for a named counter pool is shown in Figure 57.

```
STRUCTURE NAME(DFHNCLS_PRODNC1)
  SIZE(512)
  INITSIZE(256)
  PREFLIST(FACIL01,FACIL02)
```

Figure 57. Example definition of a coupling facility list structure for named counters

When you have updated the CFRM new policy with the new structure definition, activate the policy using the MVS command:

```
SETXCF START,POLICY,POLNAME=poli c yname,TYPE=CFRM.
```

A list structure can be allocated with an initial size and a maximum size, as specified by INITSIZE and SIZE respectively in the CFRM policy definition. All structure sizes are rounded up to the next multiple of 256KB at allocation time. Provided that space is available in the coupling facility, you can use the MVS SETXCF command to increase the structure size dynamically from its initial size towards its maximum size, making the new space available immediately to any currently active servers. If too much space is allocated, you can reduce the structure size to free up coupling facility storage for other purposes (which may take some time if the coupling facility has to move existing data out of the storage which is being freed). Note that if the size is altered in this way, you should also update the INITSIZE parameter in the policy to reflect the new size, so that the structure will not revert to its original size if it is subsequently recreated or reloaded.

# A coupling facility structure contains not only stored data but also the information  
# needed to manage and access that data, in a similar way to a key-sequenced data  
# set. The amount of internal control information depends on the level of functionality  
# and performance of the coupling facility control code for the current CFLEVEL, and  
# might increase for a higher CFLEVEL. For more information see “Coupling facility  
# storage management” on page 424. The space required for a named counter pool  
# depends on the number of different named counters you need, but the minimum  
# size should be enough for most needs. A minimum-size structure of 256KB can hold  
# hundred of named counters (as of CFLEVEL=14).

# For an accurate estimate of storage requirements for a list structure, use the IBM  
# CFSizer tool at <http://www-1.ibm.com/servers/eserver/zseries/cfsizer/>. The CFSizer  
# tool is a web-based application that returns structure sizes based on the latest level  
# of the coupling facility. If you enter the number of counters you require, the tool  
# calculates the size of a structure that would be sufficiently large to contain at least  
# the number of counters you specified. However, for practical operation, a  
# reasonable proportion of free space must be available, not only to minimize the risk  
# of the structure becoming full but also in order to avoid you receiving low space  
# warning messages. You should aim to utilize no more than approximately 75% of  
# the structure size. When you estimate the maximum number of counters that you  
# require, increase that number by a third to include the free space in the calculation.

Note that defining the CFRM policy statements for a list structure does not actually create the list structure. The structure is created the first time an attempt is made to connect to it, which occurs when the first named counter server that refers to the corresponding pool is started.

**Note:** Before you attempt to start a named counter server make sure you have defined and started the authorized cross-memory (AXM) server environment (see Chapter 23, “Authorized cross memory (AXM) system services,” on page 367).

---

## Defining and starting a named counter server region

You activate a named counter pool in an MVS image by starting up a named counter server region for that pool. You can start the server as a started task, started job, or as a batch job.

### The server region program, DFHNCMN

The named counter server region program is called DFHNCMN and must be run from an APF-authorized library. DFHNCMN is supplied in the CICS authorized library, CICSTS31.CICS.SDFHAUTH.

#### **SYSIN and SYSPRINT DD statements**

The server reads its initialization parameters from a SYSIN data set, and writes messages and statistics to a print file, for which it requires a SYSPRINT DD statement.

#### **Startup parameters**

The named counter pool server requires some parameters in the start-up JCL. You can specify these in the PARM string, or in the SYSIN data set, or in a combination of both. When a parameter is specified in both places, the PARM value overrides the SYSIN value (because the PARM can be overridden on the MVS START command).

The most important parameter is the pool name, which is mandatory. Among other things, the pool name is used to form, with the prefix DFHNC, the server name (giving DFHNC.*poolname*).

The easiest way to ensure that all pool-related parameters are consistent across MVS images is to use the same SYSIN parameter data set (or an identical copy of it) for all servers accessing the same pool, and to specify in the PARM field any parameters that vary between servers.

For details of all the parameters, see “Named counter server parameters” on page 412.

**Named counter server REGION parameter:** Use the JCL REGION parameter to ensure that the named counter server region has enough storage to process the maximum number of named counter requests that can be executing concurrently.

The named counter server typically uses less than one megabyte of storage above 16MB and less than 20KB below 16MB.

During server initialization, the server acquires all the available storage above 16MB, as determined by the REGION parameter, then releases 5% of it for use by operating system services. It also acquires 5% of the free storage below 16MB for use in routines that require 24-bit addressable storage.

After initialization, the server uses AXM page allocation services to manage its storage. Server statistics indicate how much storage is actually allocated and used within the storage areas above and below 16MB, which are called AXMPGANY and AXMPGLOW in the statistics.

If a task in the server region or a cross-memory request runs out of storage, this is likely to result in AXM terminating that task or request using a simulated abend with system completion code 80A to indicate a GETMAIN failure. Although the server can usually continue processing other requests in this case, running out of storage in a critical routine can cause the server to terminate. Therefore, it is best to ensure that the REGION size is large enough to eliminate this risk.

## Named counter server JCL example

```
//MVS $n$ NC1 JOB ...
//NCSEVER EXEC PGM=DFHNCMN,REGION=32M,TIME=NOLIMIT   Named counter server
//STEPLIB DD   DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR   Authorized library
//SYSPRINT DD  SYSOUT=*                               Messages and statistics
//SYSIN DD *
POOLNAME=MVS $n$ NC1                                     Pool name
/*
```

**Note:** You are recommended to specify TIME=NOLIMIT. The server task remains in a wait during most normal processing, because server processing is performed under the client CICS region TCB. If you omit the TIME subparameter, your server job could fail with abend S522 (wait limit exceeded), depending on the JWT value specified in the SMFPRMxx member of SYS1.PARMLIB.

Figure 58. Sample JCL to start a named counter server address space

## Named counter server parameters

Parameters are specified in the form KEYWORD=*value*, where keywords can optionally be specified in mixed case to improve readability. If you specify more than one parameter in the PARM field or on the same SYSIN input line, the parameters must be separated by a comma. Any text following one or more spaces is taken as a descriptive comment. Any parameter line which starts with an asterisk or a space is assumed to be a whole line comment.

You can enter some parameter keywords in more than one form, such as in abbreviated or truncated form.

The main parameters are listed on the server print file during start-up.

### Pool name parameter

This parameter, POOLNAME, is always required:

**POOLNAME=*name***

specifies the 8-character name of the named counter pool. This is appended by the server to the prefix DFHNC to create its own server name, as in DFHNC.*poolname*, and also to the prefix DFHNCLS\_ to create the name of the coupling facility list structure, as in DFHNCLS\_*poolname*.

This parameter is valid only at server initialization, and must always be specified.

This keyword can be abbreviated to **POOL**.

### Statistics parameters

Use the following parameters to specify server statistics options:

**ENDOFDAY**={00:00|*hh:mm*}

specifies the time of day, in hours and minutes, when the server is to collect and reset end-of-day statistics.

**Note:** If the STATSOPTIONS parameter specifies NONE, the server still writes end-of-day statistics to the print file.

The valid range of times is from 00:00 to 24:00.

This keyword can be abbreviated to **EOD**.

**STATSINTERVAL**={03:00|*hh:mm*}

specifies the statistics collection interval, in the range 1 minute to 24 hours. This parameter is ignored if the STATSOPTIONS parameter specifies NONE.

The time interval can range from 00:01 to 24:00.

This keyword can be abbreviated to **STATSINT**.

**STATSOPTIONS**={NONE|**SMF**|**PRINT**|**BOTH**}

specifies whether the server is to produce interval statistics, and the destination for the statistics it produces.

**NONE** The server does not produce any interval statistics.

**SMF** The server produces interval statistics and writes them to the current SMF data set only.

**PRINT** The server produces interval statistics and writes them to the server's print file only.

**BOTH** The server produces interval statistics and writes them to the current SMF data set and to the server's print file.

This keyword can be abbreviated to **STATSOPT**.

### Automatic restart manager (ARM) parameters

During server initialization, the server unconditionally registers with ARM except when the server program is invoked with either the UNLOAD or the RELOAD functions. The server will not start if the registration fails.

Use the following parameters to override default processing for the automatic restart manager:

**ARMELEMENTNAME**=*elementname*

specifies the automatic restart manager element name, up to 16 characters, to identify the server to ARM for automatic restart purposes. The permitted characters for the element name are A to Z 0-9 \$ # @ and the underscore symbol (\_).

The default identifier is of the form DFHNC*nn*\_*poolname*, where NC represents the server type, *nn* is the &SYSCClone value for the system (which can be either one or two characters), and *poolname* is the name of the pool served by the server.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMENT** or **ARMELEMNAME**.

**ARMELEMENTTYPE**=*elementtype*

specifies the automatic restart manager element type, up to 8 characters for use in ARM policies as a means of classifying similar elements. The permitted characters for the element type are A to Z 0-9 \$ # and @.

The default element type is SYSCICSS.

This parameter is only valid at server initialization.

This keyword can be abbreviated to **ARMELEMTYPE**.

### List structure parameter

These parameters specify list structure attributes. They are used only for the initial allocation of resources when the pool list structure is being created, which occurs the first time you start a server for a pool.

**POOLSIZE**={0|*number*}

specifies the initial amount of coupling facility storage to be allocated for the pool list structure, expressed as kilobytes in the form *number*K, megabytes in the form *number*M or gigabytes in the form *number*G.

**0** The special value 0 means that the server is to obtain an initial allocation using the parameters specified in the CFRM policy. If the CFRM policy specifies an INITSIZE value for the structure, this determines the initial allocation, otherwise the maximum SIZE value is allocated.

*number*

A non-zero value specifies an initial amount of storage to be allocated, overriding the INITSIZE parameter in the CFRM policy. This value is generally rounded up by MVS to the next multiple of 256K. The valid range of values is 1K to 2G, but should not be greater than the value specified on the SIZE parameter.

It is generally preferable to omit this parameter, and specify the structure size using the INITSIZE parameter in the CFRM policy. The POOLSIZE option can, however, be useful if the structure is being reallocated or reloaded, and the CFRM policy has not been updated to reflect the required size.

**Note:** If the value is greater than the value specified on the CFRM SIZE parameter, the server POOLSIZE parameter is ignored and the initial allocation is based on the parameters specified in the CFRM policy.

This parameter is valid only at server initialization and is only used when the structure is first allocated.

### Debug trace parameters

These parameters are provided only for intensive debug tracing.

Using these options in a production environment could have a significant impact on performance and cause the print file to grow very rapidly, using up spool space.

Trace messages from cross-memory requests can be lost if they are generated faster than the trace print subtask can print them. In this event, the trace only indicates how many messages were lost.

**CFTRACE**={OFF|ON}

specifies the coupling facility interface debug trace option.

**OFF** Coupling facility interface debug trace is disabled.

**ON** Coupling facility interface debug trace produces trace messages on the print file, indicating the main parameters to the coupling facility request interface, and the result from the IXLLIST macro.

This keyword can also be specified as **TRACECF**.

**RQTRACE={OFF|ON}**

specifies the request debug trace option.

**OFF** Request debug trace is disabled.

**ON** Request debug trace produces trace messages on the print file, indicating the main parameters on entry to each cross-memory request, and the results on exit.

This keyword can also be specified as **TRACERQ=**.

## Warning parameters

Use these parameters to modify the thresholds at which warning messages are issued when the structure becomes nearly full.

**ENTRYWARN={80|number}**

specifies the percentage of list structure entries in use at which warning messages should be first triggered.

The valid range is from 1 to 100 per cent.

**ENTRYWARNINC={5|number}**

specifies the percentage increase (or decrease) of entries in use before the next warning message should be triggered (reduced to 1 when the next increase would otherwise reach 100). After the first warning, additional messages are issued as the number of elements increases and decreases. These messages stop when the number of entries in use has fallen at least this percentage below the initial warning level.

The valid range is from 1 to 100 per cent.

---

## Controlling named counter server regions

You can issue commands to control a named counter server, using the MVS MODIFY (F) command to specify the job or started task name of the server region, followed by the server command. The general form of a named counter server command, using the short form F, is as follows:

```
F server_job_name,command  
parameters... comments
```

The commands supported by a named counter server are:

- SET
- DISPLAY
- PRINT
- STOP
- CANCEL

These commands and their options are as follows:

**SET** *keyword=operand[,keyword=operand,...]*

Change one or more server parameter values. The command can be abbreviated to **T**, as for the MVS SET command. See “The SET command options” on page 416 for details.

**DISPLAY** *keyword[=operand][,keyword[=operand],...]*

Display one or more parameter values, or statistics summary information, on the console. The valid keywords for DISPLAY are all the initialization parameters, plus an additional set described under “DISPLAY and PRINT command options” on page 416.

The command can be abbreviated to **D**, as for the MVS DISPLAY command.

**PRINT** *keyword*[=*operand*][,*keyword*[=*operand*,]...]

Produces the same output as DISPLAY, supporting the same keywords, but on the print file only.

#### **STOP**

Terminate the server normally. The server waits for any active connections to terminate first, and prevents any new connections while it is waiting. The command can be abbreviated to **P**.

**Note:** You can also use the MVS STOPcommand, which is equivalent to issuing the server STOP command through the MVS MODIFY command. The syntax of the STOP command is:

```
STOP|P [jobname.]identifier[,A=asid]
```

#### **CANCEL {RESTART={NO|YES}}**

Terminate the server immediately. You can specify whether automatic restart should be requested.

For information about CANCEL RESTART see “The CANCEL command options” on page 418.

## The SET command options

You can use the SET command to modify the following groups of server initialization parameters:

- The statistics parameters
- The debug trace parameters
- The warning parameters

See “Named counter server parameters” on page 412 for details of these keywords.

**Examples of the SET command:** The following example changes the statistics options:

```
SET STATSOPT=BOTH,EOD=21:00,STATSINT=06:00
```

## DISPLAY and PRINT command options

You can use the DISPLAY (and PRINT) commands to display the values of any initialization parameters plus some additional information.

The parameters supported by the DISPLAY and PRINT commands are as follows:

#### **ARMREGISTERED**

Shows whether ARM registration was successful (YES or NO).

#### **CONNECTIONS**

Display the jobnames and applids of the regions currently connected to the server to which the command is issued.

This keyword can be abbreviated to **CONN**.

#### **COUNTERS**

Display the names of all the named counters currently allocated in a pool.

**COUNTERS**={*name*|*generic\_name*}

Display the details of a specific named counter, or set of named counters whose names match the generic name. Generic names are specified using the wildcard characters \* (asterisk symbol) and % (per cent symbol).

If no named counter is specified, this is treated as equivalent to DISPLAY COUNTERS.

This keyword can be abbreviated to **COUNTER**.

### **DISPLAY and PRINT options for statistics summaries.**

Use the following parameters to display or print statistics:

#### **CFSTATS**

Display statistics for coupling facility interface accesses and responses from the server.

This keyword can also be specified as **CFST** or **STATSCF**.

#### **POOLSTATS**

Display usage statistics for the pool list structure as a whole. This is based on information returned by coupling facility access requests, therefore it is only as current as the most recent request made through the server to which the command is issued.

This keyword can be abbreviated to **POOLST**.

#### **STORAGESTATS**

Display main storage allocation statistics for the server address space.

This keyword can be abbreviated to **STORAGEST** or **STGST**.

### **DISPLAY and PRINT options for combined lists of information**

These keywords represent combined lists of information:

#### **PARAMETERS**

Display the main parameter values, which are:

POOLNAME  
STATSOPT  
ENDOFDAY  
STATSINTERVAL  
POOLSIZE

This keyword can be abbreviated to **PARAM** or **PARMS**.

#### **ALLPARAMETERS**

Display all parameter values, which are those listed for **PARAMETERS**, above, plus:

CFTRACE  
RQTRACE  
ENTRYWARN  
ENTRYWARNING

This keyword can be abbreviated to **ALLPARMS**.

#### **STATISTICS**

Display all available statistics.

This keyword can be abbreviated to **STAT** or **STATS**.

#### **INITIALIZED**

Display the parameters and statistics that are usually displayed when initialization is complete, which are those listed for **PARAMETERS** above, plus:

POOLSTATS  
STGSTATS

This keyword can be abbreviated to **INIT**.

## ARM

Display all ARM-related parameter values:

- ARMELEMENTNAME
- ARMELEMENTTYPE
- ARMREGISTERED

This keyword can be coded as **ARMSTATUS**.

## The CANCEL command options

You can use the CANCEL command to request automatic restart by specifying the following parameter:

### RESTART={NO|YES}

Terminate the server immediately, specifying whether or not automatic restart should be requested. The default is RESTART=NO.

If the server encounters an unrecoverable problem with the coupling facility connection, consisting either of lost connectivity or a structure failure, it cancels itself using the CANCEL RESTART=YES command. This terminates the existing connection and shuts down the server. A new instance of the server job is then started.

A server can also be restarted explicitly using either the server command CANCEL RESTART=YES or the MVS command CANCEL jobname,ARMRESTART.

You can also enter RESTART on its own for RESTART=YES, NORESTART for RESTART=NO.

## Server response to XES events

The server also responds to XES events such as an operator **SETXCF** command to alter the structure size. If the server can no longer access the coupling facility, it automatically issues a server **CANCEL** command to close itself down immediately.

---

## Deleting or emptying named counter pools

You can delete a named counter pool using the MVS **SETXCF** command to delete its coupling facility list structure:

For example:

```
SETXCF FORCE,STRUCTURE,STRNAME=DFHNCLS_poolname
```

You can delete a structure only when there are no servers connected to the pool, otherwise MVS rejects the command.

When you attempt to start a server for a pool that has been deleted (or attempt to reload the pool), it is allocated as a new structure. The newly allocated structure uses size and location attributes specified by the currently active CFRM policy.

---

## Changing the size of named counter pools

If the structure is becoming full, and the current pool size is less than the maximum, you can use the SETXCF START,ALTER command to increase the pool size.

For example:

```
SETXCF START,ALTER,STRNAME=DFHNCLS_poolname,SIZE=size
```

SIZE is expressed in kilobytes.

---

## Unloading and reloading named counter pools

You can unload, and reload, the complete contents of a named counter pool to and from a sequential data set by invoking the server program with the **FUNCTION** parameter, using the **UNLOAD** and **RELOAD** options.

You can use this function, for example, to

- Preserve the named counter pool during planned coupling facility maintenance, or
- Move the pool to a different coupling facility.

**FUNCTION={UNLOAD|RELOAD}**

Specify the function for which the server is being initialized.

### UNLOAD

Unload the entire contents of the named counter pool specified on the **POOLNAME** parameter to a sequential data set. When the unload processing has completed (normally or abnormally) the server program terminates.

The **UNLOAD** function requires a DD statement for **DDNAME DFHNCUL** describing the sequential data set to which the table pool is to be unloaded. The format of the unloaded data set is:

```
RECFM=F  
LRECL=4096  
BLKSIZE=4096
```

### RELOAD

Reload, into the named counter pool named on the **POOLNAME** parameter, a previously unloaded named counter pool.

The **RELOAD** function requires a DD statement for **DDNAME DFHNCRL**, describing the sequential data set from which the table pool is to be reloaded.

The structure is allocated, if necessary, during reloading, in which case you can use the same server parameters to control structure attributes as for normal server startup. The reload process bypasses named counters that are already found in the pool (for example, because the structure was too small and the reload job had to be restarted after using **ALTER** to increase the structure size).

**Note:** If the unloaded pool structure was altered dynamically at any time after initial allocation (by using the **SETXCF** command to increase the size), ensure that the increased size is allocated for the reloaded pool. The recommended way is to update the **INITSIZE** parameter for the structure in the current **CFRM** policy whenever you alter the structure size, and to activate the updated policy using the **SETXCF START,POLICY** command. Alternatively, you can specify the required pool size in the **POOLSIZE** parameter in the reload **JCL**.

**Note:** If you omit the **FUNCTION** parameter, the server program initializes a named counter server address space.

For the UNLOAD and RELOAD function, the server program requires exclusive use of the list structure. If the structure is currently being used by a normal server, the unload or reload attempt is rejected. Similarly, if a normal server attempts to start up while an unload or reload job is in progress, the attempt fails because shared access to the structure is not available.

You can specify all normal server parameters when unloading or reloading, but some of these (for example, statistics-related parameters) are ignored because they do not apply to unload or reload processing.

If reloading fails because it runs out of space, the resulting messages include the numbers of named counters reloaded and blocks read up to the time of the failure. You can compare these values with those in the messages from the original unload job, to determine how many more named counters remain to be loaded.

## Unload JCL example

```
//UNLDNCD1 JOB ...
//NCUNLOAD EXEC PGM=DFHNCMN          CICS named counter server program
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=*              Options, messages and statistics
//DFHNCUL DD DSN=NC1.UNLOADED.POOL,  Unloaded named counter pool
//          DISP=(NEW,CATLG),
//          SPACE=(4096,(10000,1000)) Estimated size in 4K blocks
//SYSIN DD *
FUNCTION=UNLOAD                      Function to be performed is UNLOAD
POOLNAME=PRODNC1                    Pool name
/*
```

Figure 59. Unload JCL example

## Reload JCL example

```
//RELDNCD1 JOB ...
//NCRELOAD EXEC PGM=DFHNCMN          CICS named counter server program
//STEPLIB DD DSN=CICSTS31.CICS.SDFHAUTH,DISP=SHR Authorized library
//SYSPRINT DD SYSOUT=*              Options, messages and statistics
//DFHNCRL DD DSN=NC1.UNLOADED.POOL,DISP=OLD Unloaded pool
//SYSIN DD *
FUNCTION=RELOAD                      Function to be performed is RELOAD
POOLNAME=PRODNC1                    Pool name
/*
```

Figure 60. Reload JCL example

---

## Dumping named counter pool list structures

You can use the MVS DUMP command to obtain a dump of the coupling facility list structure for a named counter pool. For information on dumping and formatting a list structure, see the *CICS Problem Determination Guide*.



## # Chapter 27. Coupling facility server operations

# The information in this section refers to all three CICS coupling facility servers, for  
# temporary storage, coupling facility data tables and named counters, unless  
# otherwise indicated. For more detailed information about any one of the coupling  
# facility servers, refer to the individual sections:

- # • Chapter 24, “Setting up and running a temporary storage server,” on page 369
- # • Chapter 25, “Setting up and running a coupling facility data table server,” on  
# page 383
- # • Chapter 26, “Setting up and running a named counter server,” on page 405

---

### # Monitoring coupling facility server messages

# The server issues various messages during execution, some of which might  
# indicate that serious problems are developing, for example that the coupling facility  
# structure is becoming full. It is important to understand the types of messages that  
# the server issues and to ensure that system status messages are monitored for  
# possible problems.

### # Server messages

# Messages that are issued by the server code itself start with the five-letter server  
# prefix (DFHXQ, DFHCF or DFHNC). These messages fall into the following groups:

- # • Operator console system status messages, which are issued by WTO (Write To  
# Operator) with routing codes 2 (operator information) and 11 (programmer  
# information), and descriptor code 4 (system status).

# These messages provide information about important status changes, primarily  
# about the beginning and end of server initialization and the beginning and end of  
# server termination, and about problems.

# Any server message that is issued in this way during normal running indicates a  
# potentially serious problem, and should not be ignored. If this process is  
# automated, a simple rule is to ignore the specific list of status messages for  
# normal initialization and termination, and to treat any other server message as a  
# warning.

# These messages can be issued either from the server address space or from a  
# client address space. If the server code that requests the message is running in  
# cross-memory mode, the message is passed back to a routine that issues the  
# WTO in primary mode in the client address space. This avoids restrictions which  
# apply to WTO messages that are issued in cross-memory mode. For example,  
# cross-memory mode WTO messages do not appear in any job log.

- # • Command responses, which are issued by WTO with routing codes 2 and 11 and  
# descriptor code 5 (immediate command response).

# These messages contain responses to server commands that are issued via the  
# system MODIFY or STOP command, for example to display statistics.

- # • Job log messages:

# These messages typically contain diagnostic information, for example details of  
# an abend or an attempted security violation. They are written to the job log, by  
# WTO with routing code 11 (programmer information).

- # • Trace and statistics messages:

# These messages are written only to the server SYSPRINT file.

# All messages that are issued by the server code, whether they are running under  
# the server address space or in cross-memory mode from the client address space,  
# are also copied to the server SYSPRINT file.

## # AXM messages

# The AXM environment code issues operator messages from the server and client  
# address spaces and from the master address space during AXM system services  
# initialization. These messages are issued using WTO with routing codes 2 (operator  
# information) and 11 (programmer information, not used when running in the master  
# address space), and descriptor code 4 (system status). AXM message numbers are  
# of the form "AXMxxnnns", where the first five characters "AXMxx" are the name of  
# the module issuing the message, "nnnn" is the numeric part of the message  
# number, and "s" is the suffix letter. The suffix letter is "I" if the message is a routine  
# informational message, or is omitted if the message indicates an error. The suffix  
# letter can be used by automation tools to distinguish routine informational messages  
# from error situations.

# AXM messages that are issued from the server environment (via AXM run-time  
# environment routines linked with the server load module), are copied to the server  
# SYSPRINT file as well. AXM also writes informational messages to the SYSPRINT  
# file. These contain information such as initialization information and closedown  
# statistics for storage management, and the main procedure entry point of the server  
# module for diagnostic purposes.

#

---

## # Coupling facility storage management

# The type of coupling facility structure that CICS uses for its temporary storage data  
# sharing pools, coupling facility data table pools and named counter pools is a keyed  
# list structure. This contains an array of numbered lists, each of which is like a keyed  
# file, containing entries with 16-byte keys. Each entry has a fixed prefix area  
# containing its key and other control information that includes an "adjunct area" of 64  
# bytes for program use, followed by a chain of up to 128 256-byte data elements.  
# This allows a maximum data size of 32K bytes. For named counters, only the prefix  
# area is used, with no data elements, and there is only one list. Storage for entry  
# prefixes, known as entry controls, and data elements is allocated from two storage  
# pools within the structure, which are shared between all lists in the structure.

# The storage in a list structure can be divided into two basic types: fixed controls  
# and variable controls. Storage for fixed controls is preallocated at a fixed size for  
# the life of the structure. It contains structure control information, including data  
# buffer space, as well as an array of list headers, up to the maximum number of lists  
# defined when the structure was created. For CICS, it consists of a small number of  
# lists used for CICS internal purposes, and the value specified for the parameters  
# **MAXQUEUES** or **MAXTABLES**.

# Variable controls are partitioned into storage areas for entry controls, one per entry,  
# and for data elements. You can dynamically adjust this partitioning by altering the  
# ratio of entries to elements, converting storage for one type to the other type. CICS  
# automatically issues a request to alter the ratio when one type of variable storage is  
# running out but there is enough storage of the other type. You can dynamically alter  
# the total size of the storage for variable controls by changing the size of the  
# structure. This must be within the range of sizes that are defined for the structure in  
# the active CFRM policy, which are set at the time that the structure is created. You  
# can change the structure size by using the system operator command SETXCF

# ALTER,SIZE. Alternatively the operating system can change the structure size,  
# depending on the automatic alter options that are specified in the CFRM policy  
# when the structure is allocated.

# The fixed controls include enough internal control areas to manage the maximum  
# number of elements and entries that can exist in the structure, given the maximum  
# structure size and the range of possible ratios of entries to elements, and an array  
# of list headers to handle the maximum number of lists that can exist in the  
# structure. As neither the maximum size nor the maximum number of lists can be  
# increased without reallocating the structure, this means that you must be careful to  
# specify large enough values when the structure is first allocated. However, if you  
# specify a relatively large maximum size or number of lists, a large amount of  
# storage is preallocated for fixed controls; so the initial size of the structure needed  
# to store a given amount of data will be significantly larger than it would be with less  
# generous allowance for expansion.

# Within the CICS pool structures, each queue item, data table record or named  
# counter normally occupies one entry in the structure, together with the appropriate  
# number of data elements. Additional entries are used by CICS for internal control  
# purposes to maintain the index of currently defined queues or data tables, as well  
# as tracking which lists are currently in use and which previously used lists are now  
# free and available for reuse. Note that even after all queues or data tables in a pool  
# have been deleted, some entries in the control lists might remain in use.

# The amount of storage required for internal control information depends on the level  
# of functionality and performance of the coupling facility control code for the current  
# CFLEVEL, and may increase for a higher CFLEVEL. The easiest way to calculate  
# storage requirements for the CICS pool structures is to use the web-based IBM  
# CFSizer tool at <http://www-1.ibm.com/servers/eserver/zseries/cfsizer/>. This tool  
# prompts for parameters that describe the amount of information to be stored in the  
# structure in terms of the corresponding CICS resources, converts this information to  
# numbers of lists, entries and elements and then communicates with a coupling  
# facility at a current CFLEVEL to determine the amount of storage required to store  
# the information with the specified amount of free space and room for expansion. For  
# more information about calculating storage requirements, see “Approximate storage  
# calculations” on page 38.

---

## # **Managing the pool structure**

# It is important to watch out for signs that the pool structure is becoming full, as this  
# might have a serious impact on all the applications that are using the pool.

### # **Monitoring pool structure usage levels**

# Use the DISPLAY POOLSTAT server command to display the current usage level of  
# the pool structure. The DISPLAY POOLSTAT command produces messages  
# DFHXQ0432I, DFHF0432I or DFHNC0432I. The most important information in  
# these messages is the maximum percentage of lists, entries, and elements, used  
# during the current statistics interval. For the named counter server, message  
# DFHNC0432I only shows the number of entries because there is always one list  
# and no elements.

### # **Operator messages reporting on pool structure usage**

# Messages to the operator, for example DFHXQ0411I and DFHXQ0412I, are issued  
# when threshold levels are reached for the numbers of entries or elements used.  
# Further messages are issued if the pool becomes full. You can set up automated

# operations processes to watch for these messages and alert your operators to the  
# situation, or take corrective action, if necessary.

# You can use the MVS operator command SETXCF ALTER,START with an  
# increased SIZE option, to expand the structure, if the structure has not yet reached  
# the maximum size that is defined in the CFRM policy.

### # **Use of CFRM automatic ALTER to increase pool structure size**

# The coupling facility resource management (CFRM) policy can specify the keyword  
# ALLOWAUTOALT(YES). This allows the operating system to issue an ALTER  
# command automatically when the structure is near to becoming full, to increase its  
# size or to adjust the ratio of elements to entries. The threshold at which this  
# happens is specified by the FULLTHRESHOLD keyword in the policy. The default  
# threshold is 80%, which is the same as the default threshold at which the server  
# itself issues an automatic ALTER command to optimize the ratio of entries to  
# elements. The server's automatic ALTER process is more sophisticated, because it  
# takes into account the peak usage of the structure within the current statistics  
# interval, rather than just the current usage. Therefore it is best to ensure that the  
# server's automatic ALTER process is triggered first, by making sure that the  
# percentage values of the server's ENTRYWARN and ELEMENTWARN parameters  
# are at least 5 percent less than the percentage value of the CFRM  
# FULLTHRESHOLD keyword.

### # **Using system-managed rebuild to increase pool structure size**

# If the structure has not many entries or elements left, but it has already reached its  
# maximum size, you can still use system-managed rebuild to expand it dynamically  
# without closing down the servers, if all the systems using the structure are at a level  
# which supports this function. First update the CFRM policy to increase the size to  
# the required value, and then activate the updated policy using SETXCF  
# START,POLICY. After this, you can rebuild the structure. The rebuild allocates a  
# new instance of the structure that uses the updated policy, copies across the  
# existing data, and then discards the old instance.

### # **Increasing the number of data lists**

# If the number of data lists specified via the MAXQUEUEES or MAXTABLES server  
# parameter is too small, an attempt to allocate a new data list will fail with message  
# DFHXQ0443 or message DFHCF0443. There is no way to increase the number of  
# lists without deleting and recreating the structure, which necessitates closing down  
# all of the servers temporarily. You cannot use system-managed rebuild to increase  
# the number of data lists because it copies this number from the existing structure.

# You can preserve existing data by using the server program to unload it to a  
# sequential file, and then using SETXCF FORCE to delete the existing structure. You  
# can then use the server program again to reload the data, and allocate a new  
# structure with the appropriate MAXQUEUEES or MAXTABLES parameter.

### # **Deleting or emptying the pool structure**

# If the pool structure is no longer required, or all of the data in the structure is to be  
# discarded, you can delete a pool by closing down all the servers for that pool, and  
# then using the SETXCF FORCE command to delete the structure. If the server is  
# subsequently started again for the same structure name, an empty structure will be  
# created using the information in the active CFRM policy and the server initialization  
# parameters.

---

## # **Server connection management**

### # **Establishing server connections**

# A client CICS region establishes a cross-memory connection to a server the first  
# time a request refers to the pool for that server. A single connection is established  
# to each server, regardless of the number of tables, queues or counters that are  
# accessed in the pool.

# For coupling facility data table and temporary storage queue servers, a  
# multi-threaded asynchronous connection is established. This allows requests to be  
# overlapped up to a fixed maximum number of concurrent requests. Requests that  
# exceed this maximum number are queued within the CICS region until a request  
# thread becomes available.

# For named counter servers, requests are processed synchronously using a  
# single-threaded interface, and only one request can be active at a time for a given  
# client region.

### # **Terminating server connections**

# Each connection has a client side and a server side. If either side is terminated  
# separately, the connection is no longer usable, although the other side might not be  
# terminated until some time later.

# A connection normally remains active until CICS is closed down. The connection is  
# closed automatically as part of the resource management termination processing for  
# the CICS quasi-reentrant TCB. During resource management termination, the  
# connection termination routine on the client side issues a cross-memory call to the  
# server to terminate the connection on the server side as well.

# If you wish to close down a server after all the CICS regions that are using it have  
# terminated, for example when you are preparing to close down the system, first  
# quiesce the server and then terminate it using the server STOP command (or the  
# equivalent MVS STOP command). Use of the STOP command prevents any new  
# CICS regions from connecting to the server and terminates the server as soon as  
# all the CICS regions that are using it have been terminated.

# If you need to close down a server immediately, while CICS regions are still  
# connected to it, use the server CANCEL command, because the normal server  
# STOP command, (or the MVS STOP command) is not effective until the  
# connections have been terminated. You can also use the MVS CANCEL command  
# to terminate the server, but this prevents the server from going through normal  
# closedown processing.

# Note that CICS cannot terminate the client side of the connection automatically.  
# CICS has no way of knowing that the server wishes to close down, because  
# although the connection allows CICS to make cross-memory calls to the server, it  
# does not provide any means for the server to notify CICS of asynchronous events.  
# At present, CICS does not provide any means of terminating a connection on  
# demand except in the case of the named counter server CALL interface which  
# provides a FINISH function for this purpose, but this function is primarily for batch  
# use.

# If you terminate the server with CANCEL, the server side of each connection is  
# terminated immediately, but the client side is not affected. The next request from

# CICS to use the original connection fails and CICS tries to establish a new  
# connection instead. This succeeds if the server has been restarted. However, CICS  
# does not close the old connection explicitly, so when it eventually terminates,  
# messages are produced not only for the termination of any active connection, but  
# also for the termination of any previous connections to the same server.

## # Failed server connections

# If CICS terminates abruptly, without going through the normal resource manager  
# termination processing for the quasi-reentrant TCB, for example because of a  
# FORCE command or system completion code 40D, an end-of-memory resource  
# manager routine cleans up the client side of the connection. Because this routine  
# does not run in the CICS region itself, it cannot use the cross-memory connection  
# to notify the server. Therefore, if CICS terminates without carrying out the normal  
# resource manager termination processing for the quasi-reentrant TCB, the server  
# side of the connection might remain active. For coupling facility data tables this  
# might cause problems because the server does not allow the original CICS region  
# to resynchronize after restart if its APPLID is already in use.

# From time to time the server checks the client status for each connection, and  
# cleans up the server side of the connection if the client has gone away. This check  
# is done once a minute, and it is also triggered each time a new connection is to be  
# established. Therefore, any connections which have failed are normally cleaned up  
# before the new connection attempts to resynchronize. Because the clean-up  
# processing that terminates the server side of a connection is asynchronous, and  
# might take one or two seconds, the clean-up processing might not be finished in  
# time for the resynchronization from the original CICS region to succeed  
# immediately, but if the resynchronization does not succeed on the first attempt, it  
# should succeed when it is next retried.

## # Restarting a server

# All three types of CICS data-sharing server (temporary storage, coupling facility  
# data tables, and named counters) support automatic restart using the services of  
# the Automatic Restart Manager (ARM). The servers also have the ability to wait  
# during start-up, using an Event Notification Facility (ENF) exit, for the coupling  
# facility structure to become available if the initial connection attempt fails.

# The server normally registers at start-up with ARM, so that it will be restarted  
# automatically if it fails, subject to any rules in the installation ARM policy. If ARM  
# registration fails for any reason except for ARM being unavailable, the server cannot  
# be started. If ARM is unavailable, the server starts normally but has to be restarted  
# manually if it fails.

# The servers recognize the ARM return code that indicates that the ARM couple data  
# set has not been formatted for the current MVS system, as being equivalent to ARM  
# being unavailable.

# A server does not start if registration fails with return code 8 or above.

# When a server starts up, if it is unable to connect to its structure because of some  
# environmental error such as a structure failure, it automatically waits for the  
# structure to become available, using the Event Notification Facility (ENF) to watch  
# for events relating to its structure. This wait occurs before the cross-memory  
# interface is enabled, so the server is not visible to client regions at this time and will  
# simply appear to be unavailable. While it is waiting, the server can be cancelled  
# using the MVS CANCEL command if it is no longer required.

# If the server is running normally, but the coupling facility interface reports a loss of  
# connectivity or a structure failure, the server immediately terminates itself. This  
# disconnects it from the coupling facility, and terminates the server side of any  
# current cross-memory connections from client regions. The server will normally be  
# restarted immediately by the ARM, but will continue to be unavailable to client  
# regions until the coupling facility structure is available again (possibly as a new  
# empty instance of the structure).

# An abrupt coupling facility failure such as a power failure may result in a loss of  
# connectivity indication even though the structure has failed, because the operating  
# system cannot determine the state of the structure in that case. This could prevent  
# a new structure from being allocated until the operating system can determine the  
# status of the existing structure, for example after the failed coupling facility has  
# been successfully restarted. If it is certain that the old structure has been lost, but  
# the system has not yet recognized the fact, the operator may be able to save some  
# time by issuing the SETXCF FORCE command to delete the old structure, allowing  
# the system to go ahead and create a new instance of the same structure in a  
# different coupling facility.

# You can find more information about automatic restart of coupling facility servers in  
# the *CICS Recovery and Restart Guide*



---

## Chapter 28. CICS server support for system-managed processes

This chapter describes the system-managed processes supported by the CICS data sharing servers. The three servers, temporary storage data sharing, coupling facility data tables, and named counters, support the following system-managed processes for coupling facility list structures:

- “System-managed list structure rebuild”
- “System-managed list structure duplexing” on page 433

Each of these system-managed facilities, and CICS support for them, is described in the following topics.

---

### System-managed list structure rebuild

System-managed rebuild allows OS/390 or z/OS to manage the moving of coupling facility structures used for shared temporary storage, coupling facility data tables, and named counter server pools, without recycling the CICS system using them. Before this, you could move a structure by using server functions to UNLOAD to a sequential data set and then RELOAD elsewhere from the data set, but the switch caused errors in CICS such that restart was recommended, which in some situations was an unacceptable outage.

System-managed rebuild rebuilds the contents of a coupling facility list structure to a new location. The only impact when rebuild occurs is that requests are temporarily suspended within MVS during the few seconds or tens of seconds needed for rebuild processing. The system-managed rebuild process rebuilds only from one structure to another. Therefore, it is not applicable when the original structure has been lost or damaged. It is very useful for planned maintenance and is used for recovery purposes where connectivity to the structure has been lost from one or more systems but at least one system still has access to the original structure. For a loss of connectivity, the system does not initiate a system-managed rebuild automatically, regardless of connectivity options in the policy, but a rebuild can be requested using an operator command.

Any pending requests are simply made to wait during system-managed rebuild. Apart from the time delay, the application should not notice anything unusual when a system-managed rebuild occurs. For more information about system-managed rebuild, see *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771.

CICS supports the MVS system-managed rebuild facility, provided that the SUSPEND=FAIL option of the IXLCONN macro is available (SUSPEND=FAIL was introduced by OS/390 APAR OW39892). The CICS servers detect automatically whether this support is available. If SUSPEND=FAIL support is available, a server connects to its list structure specifying the ALLOWAUTO=YES option of the IXLCONN macro. If SUSPEND=FAIL is not available, a server connects with ALLOWAUTO=NO, and the system-managed rebuild facility is not used.

When a server is active, you can use the MVS operator DISPLAY XCF,STR command to display the connection details in message IXC360I.

For example, to look at the connection details for structure DFHXQLS\_PRODTSQ1, use the following command:

```
Display XCF,STR,STRNAME=DFHXQLS_PRODTSQ1,CONNNAME=ALL
```

To look at the connection details for coupling facility data tables structure DFHCFLS\_DTPOOL1, use the following command:

```
Display XCF,STR,STRNAME=DFHCFLS_DTPOOL1,CONNAME=ALL
```

To look at the connection details for the named counter structure DFHNCLS\_PRODNC1, use the following command:

```
Display XCF,STR,STRNAME=DFHNCLS_PRODNC1,CONNAME=ALL
```

Using the above command, the resulting IXC360I message output contains the following lines (under the CONNECTION information section), indicating that system-managed rebuild is enabled for the connection:

```
ALLOW AUTO      : YES  
SUSPEND         : FAIL
```

## TS data sharing and CFDT servers

For temporary storage data sharing and coupling facility data tables, any pending requests are simply made to wait during system-managed rebuild. Apart from the time delay, the application should not notice anything unusual when a system-managed rebuild occurs.

### Timeout considerations

The wait-time for a system-managed rebuild is expected to vary from a few seconds for a small structure to a few tens of seconds for a large structure. This means that transactions can be purged by DTIMOUT, or by an operator command, while waiting on the rebuild.

To minimize the risk of unnecessary problems caused by timeouts during syncpoint processing, the CICS wait exit for CFDT unit of work control functions specify that the wait is not purgeable.

**Note:** Making the wait for CFDT unit of work control functions non-purgeable does not apply to CICS TS 1.3 regions connected to a later level of the CFDT server that supports system-managed rebuild. In this case, if a transaction makes any recoverable changes to a coupling facility data table before being purged, it will probably go into a wait again during syncpoint processing. If this happens, it should be left waiting until the rebuild completes, because any further attempt to purge it could result in the UOW being shunted as having failed during syncpoint. If the UOW is shunted in these circumstances, it will require manual intervention to retry syncpoint processing and complete the UOW when the structure becomes available again.

## Named counter server

Requests issued using the CALL interface to the named counter server are not made to wait during rebuild, but instead the server returns a new environment error return code with value 311, under the NC\_ENVIRONMENT\_ERROR category.

An EXEC interface request during rebuild waits using a timer wait and retry loop which can be interrupted by an operator purge command but is not eligible for DTIMOUT, since the wait is definitely known not to be caused by any form of deadlock.

## Compatibility considerations

The named counter client region interface module DFHNCIF normally resides in a linklist library and should be at the highest level of CICS TS in use within an MVS image.

If a CICS region is running a CICS TS 2.2 or later level of DFHNCIF (from the link list) but the EXEC CICS interface is from CICS TS 2.1 or earlier, the named counter server does not wait and retry during rebuild, but instead returns INVREQ with RESP2 code 311.

If a program running in a region that is using a CICS TS 2.1 or earlier DFHNCIF issues a request to a CICS 2.2 or later level of the server during a rebuild, CICS returns error code 301, UNKNOWN\_ERROR. For the EXEC interface, this is mapped to an INVREQ with RESP2 code 301.

---

## System-managed list structure duplexing

System-managed coupling facility duplexing provides a general-purpose, hardware-assisted mechanism for duplexing coupling facility structure data. It is a robust mechanism that provides recovery from failures such as loss of a single structure or coupling facility, or from loss of connectivity, by a rapid switch to the other structure in a duplex pair.

Unlike system-managed rebuild, which can rebuild only from one structure to another, and is not applicable when the original structure has been lost or damaged, system-managed duplexing creates and maintains a duplexed copy of a structure in advance of any failure. It is largely transparent to the users of the structure, and is available for both planned and unplanned outages of a coupling facility or structure. For example, until CICS TS 2.2, coupling facility data tables and temporary data sharing structures have contained mostly “scratch-pad” information that is not critical to recover. With duplexing, you can use the coupling facilities to store more critical information.

Transactions that are accessing a duplexed structure might experience delays, which could result in the DTIMOUT threshold being reached, when:

- A structure is quiesced by MVS while duplexing is being established.
- A structure is quiesced as a result of an operator command to stop or start duplexing.
- A structure is switched to the secondary structure or back to the primary structure.

A newly-started data sharing server is not allowed to connect to a structure during any phase of a duplexing rebuild until the duplex established phase is reached.

System-managed duplexing is built on the system-managed rebuild function. However, whereas system-managed rebuild requires OS/390 with APAR OW39892, system-managed duplexing requires z/OS Version 1 Release 2 with an enabling APAR. You also need to specify the DUPLEX option, in the CFRM policy information for the structure, as either DUPLEX(ENABLED) or DUPLEX(ALLOWED). If you specify DUPLEX(ENABLED), MVS will initiate and attempt to maintain a user-managed duplexing operation for the structure. If you specify DUPLEX(ALLOWED), an operator can start duplexing, using the SETXCF START,REBUILD,DUPLEX command. You also need to ensure that coupling facilities that are taking part in duplexing operations communicate with one another through a “peer link”.

For more information about system-managed duplexing, see *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771, and *OS/390 MVS Setting Up a Sysplex*, GC28-1779.

---

## Bibliography

---

### 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 Release Guide*, GC34-6421  
*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 Program Directory*, GI10-2586  
*CICS Transaction Server for z/OS Release Guide*, GC34-6421  
*CICS Transaction Server for z/OS Migration from CICS TS Version 2.3*, GC34-6425

*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.

<i>Designing and Programming CICS Applications</i>	SR23-9692
<i>CICS Application Migration Aid Guide</i>	SC33-0768
<i>CICS Family: API Structure</i>	SC33-1007
<i>CICS Family: Client/Server Programming</i>	SC33-1435
<i>CICS Transaction Gateway for z/OS Administration</i>	SC34-5528
<i>CICS Family: General Information</i>	GC33-0155
<i>CICS 4.1 Sample Applications Guide</i>	SC33-1173
<i>CICS/ESA 3.3 XRF Guide</i>	SC33-0661

---

## Books from related libraries

### DATABASE 2 (DB2)

- *DB2 for OS/390 and z/OS Administration Guide* , SC26-9931
- *DB2 for OS/390 and z/OS Application Programming and SQL Guide* , SC26-9933

### MVS

- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752
- *OS/390 MVS Installation Exits*, SC28-1753
- *OS/390 MVS Conversion Notebook*, GC28-1747
- *OS/390 MVS System Commands*, GC28-1781
- *OS/390 MVS Diagnosis: Tools and Service Aids*, SY28-1085
- *OS/390 TSO/E System Programming Command Reference*, SC28-1972
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779
- *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771
- *OS/390 MVS Programming: Sysplex Services Reference*, GC28-1772
- *OS/390 MVS IPCS User's Guide*, GC28-1756
- *OS/390 Parallel Sysplex Application Migration*, GC28-1863.

## Java

- *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide, SC34-6358*

---

### 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

## Special characters

.END control keyword 273

## A

ACF/NCP 20  
ACF/VTAM 20  
active delay interval for XRF 220  
activity keypoint frequency (AKPFREQ) 167  
ADI, system initialization parameter 164  
ADYN, dynamic allocation transaction 119  
AIBRIDGE, system initialization parameter 164  
AICONS, system initialization parameter 164  
AIXIT, system initialization parameter 165  
AILDELAY, system initialization parameter 165  
AIQMAX, system initialization parameter 166  
AIRDELAY, system initialization parameter 166  
AKPFREQ, system initialization parameter 167  
alternate delay interval for XRF 164  
AMP parameter for specification of the GCD 348  
APPL statement, VTAM VBUILD application  
  identifier 167  
APPLID, system initialization parameter 167  
ARMREGISTERED, named counter server 416  
ASLTAB (VTAM macro) 8  
AUTCONN, system initialization parameter 168  
authorized libraries 345  
AUGHTYPE DB2 parameter 263  
AUTODST, system initialization parameter 168  
autoinstall  
  for VTAM-connected terminals 4  
  installing terminal resource definitions 7  
  terminal definitions 7  
automatic dynamic storage tuning  
  AUTODST 168  
automatic installation (autoinstall) 3  
automatic start 239, 279  
AUTORESETTIME, system initialization parameter 168  
autostart override record 87  
auxiliary storage trace 169  
auxiliary temporary storage data set 35, 37, 347  
  control interval size 36  
  job control statement for CICS execution 37  
  job control statements to define 35  
  number of control intervals 37  
  space considerations 36  
auxiliary trace data sets 95  
  job control statements for CICS execution 97  
  job control statements to allocate 96  
  space calculations 97  
auxiliary trace utility program, DFHTU640 98  
AUXTR, system initialization parameter 169  
AUXTRSW, system initialization parameter 169

## B

backout exit programs 247  
backout of resources at emergency restart 247  
batching requests 214  
BDAM data sets  
  creating and loading 116  
  opening and closing 120  
BMS (basic mapping support)  
  BMS system initialization parameter 169  
  page-chaining command character string 221  
  page-copying command character string 221  
  page-purging command character string 221  
  page-retrieval command character string 221  
  PGCHAIN, BMS CHAIN command 221  
  PGCOPY, BMS COPY command 221  
  PGPURGE, BMS PURGE command 221  
  PGRET, BMS RETRIEVAL command 221  
  PRGDLY, BMS PURGE DELAY command 223  
  purge delay time interval 223  
  selecting versions of BMS 270, 271  
  versions of BMS 170  
BMS, system initialization parameter 169  
BRMAXKEEPTIME, system initialization  
  parameter 171  
BSAM devices 10  
  DD statements for 10, 342  
  with START requests 12  
BTS data set, DFHLRQ 351  
buffers and strings, VSAM 249, 252  
BWO (backup while open)  
  data facility data set services (DFDSS) 34  
  data facility hierarchical storage manager  
  (DFHSM) 34  
  disabling activity keypointing 33  
  introduction 31  
  restrictions 33  
  storage management facilities 33  
  storage management subsystem (SMS) 34  
  XRF considerations 33

## C

CADL command log for RDO 64, 80  
CAIL command log for RDO 80  
CANCEL command, named counter server  
  regions 418  
card reader 10  
card reader and line printer 10  
cataloged procedures  
  starting CICS as a started task 360  
catalogs 278  
CAVM (CICS availability manager)  
  CAVM control data set 29  
  CAVM message data set 29  
  DD statements in CICS startup job 107  
  DFHXRMMSG, XRF data set 110  
  DFHXRMMSG, XRF message data set 107

CAVM (CICS availability manager) *(continued)*  
 I/O error handling 110  
 JCL to define XRF message data set 107  
 job stream to define DFHXRCTL 106  
 required data sets 105  
 security of XRF data sets 110  
 space calculations 106  
 surveillance signal in the control data set 105  
 CCSO transient data destination 348  
 CDSA (CICS-key DSA) 357  
 CDSASZE, system initialization parameter 171  
 CEBT transaction 14  
 CEDA transaction 5  
 defining consoles devices 14  
 defining file resources 5  
 installing VTAM terminal definitions 7  
 recovery and backup 76  
 sharing a CSD between more than one CICS region 63  
 CEDB transaction 5, 64  
 CEDC transaction 5, 64  
 CEMT master terminal transaction 101  
 CESF GOODNIGHT transaction 12  
 CESF LOGOFF transaction 13  
 CESN transaction 16  
 CESO transient data destination 348  
 CETR, trace control transaction 96  
 CHKSTRM, system initialization parameter 171  
 CHKSTSK, system initialization parameter 171  
 CICS auxiliary trace utility program(DFHTU640) 25  
 CICS journal utility program (DFHJUP) 61  
 CICS server support for system-managed processes 431  
 CICS Web interface  
 WEBDELAY system initialization parameter 256  
 CICS-key storage 354  
 CICSSVC, system initialization parameter 172  
 CICSTS31.CICS.STEPLIB, CICS load library 345  
 CILOCK, system initialization parameter 172  
 class, monitoring 211  
 CLINTCP, system initialization parameter 172  
 close destination request limit 219  
 CLSDST destination request limit 219  
 CLSDSTP, system initialization parameter 172  
 CLT (command list table) 173  
 CLT, system initialization parameter 173  
 CMAC support, messages data set 135  
 CMDPROT, system initialization parameter 173  
 CMDSEC, system initialization parameter 174  
 COLD option  
 system initialization parameter BMS 169  
 system initialization parameter ICP 203  
 system initialization parameter START 240  
 system initialization parameter TS 252  
 COMAUGHTYPE DB2 parameter 263  
 command list table (CLT) 173  
 command logs, RDO 80  
 commands  
 CEDA command syncpoint criteria 78  
 CEDA DEFINE 7  
 CEDA INSTALL 5  
 commands *(continued)*  
 CEDA INSTALL GROUP(groupname) 7  
 CEDA LOCK 5  
 DEFINE TERMINAL CONSNAME(name) 16  
 DEFINE TERMINAL CONSOLE(number) 17  
 DFHCSDUP INITIALIZE 17  
 LIST ALL OBJECTS 65  
 MODIFY command 14  
 RDO CEDA INSTALL 85  
 REPRO, to run IDCAMS 113  
 common work area (CWA) 256  
 common work area storage key  
 system initialization parameter 182  
 CONFDATA, system initialization parameter 174  
 CONFTXT, system initialization parameter 176  
 CONSOLE, control keyword 273  
 consoles  
 CONSOLE, control keyword 273  
 DEFINE TERMINAL CONSNAME(name)  
 command 16  
 DEFINE TERMINAL CONSOLE(number)  
 command 17  
 defining a TSO user 16  
 defining to CICS 14  
 devices 14  
 entering system initialization parameters 277  
 OS/390 console as a CICS master terminal 14  
 TSO user, defining as a console device 14  
 control data set 105  
 control tables  
 defining CICS resources 5  
 copybooks  
 DFH\$TCTS 11  
 coupling facility data tables  
 defining resources for 124  
 starting a server 383  
 coupling facility list structures  
 for coupling facility data tables 124  
 CPLD transient data destination 348  
 CPLI transient data destination 348  
 CPSMCONN, system initialization parameter 176  
 CRDI command log for RDO 64, 80  
 CRLPROFILE, system initialization parameter 177  
 CSD (CICS system definition file)  
 CSD and control tables 5  
 CSDSTRNO 182  
 DD statements in CICS startup job 81  
 defining 63  
 definitions for the Japanese language feature 82  
 dynamic allocation 81  
 emergency restart and backout 78  
 GRPLIST=listname system initialization parameter 4  
 job control statements for CICS execution 81  
 job to define and initialize 65  
 making the CSD available to CICS 81  
 moving CICS tables to the CSD 82  
 protecting groups of resources 5  
 recovery and backup 76  
 sharing the CSD 68  
 space for data set 64

CSD (CICS system definition file) *(continued)*  
 using autoinstall to define VTAM-connected terminals 4  
 using CEDA INSTALL 5  
 CSDACC, system initialization parameter 177  
 CSDBKUP, system initialization parameter 178  
 CSDBUFND, system initialization parameter 178  
 CSDBUFNI, system initialization parameter 178  
 CSDDISP, system initialization parameter 178  
 CSDDSN, system initialization parameter 179  
 CSDFRLOG, system initialization parameter 179  
 CSDJID, system initialization parameter 180  
 CSDL command log for RDO 64, 80  
 CSDLSRNO, system initialization parameter 181  
 CSDRECOV, system initialization parameter 181  
 CSDSTRNO, system initialization parameter 182  
 CSECT operand of system initialization parameter TYPE 164  
 CSFL command log for RDO 64, 80  
 CSFU, CICS file utility transaction 120, 123  
 CSKL command log for RDO 64, 80  
 CSPL command log for RDO 64, 80  
 CSRL command log for RDO 64, 80  
 CSSL transient data destination 348  
 cushion storage 358  
 CWA (common work area) 256  
 CWAKEY, storage key for the CWA 354  
 CWAKEY, system initialization parameter 182  
 CXRF queue 46  
 CXRF transient data queue 46  
 DD statements, DISP operand 47  
 DISP operand of DD statements 47

## D

DAE, system initialization parameter 183  
 data facility data set services 34  
 data facility hierarchical storage manager (DFHSM)  
 backup while open 31  
 data interchange program (DIP) 184  
 data sets  
 actively shared 29  
 allocation 30  
 allocation and dispositions (XRF) 29  
 auxiliary temporary storage 35, 37  
 auxiliary trace 95  
 BDAM 116  
 catalog data sets 85, 91  
 CAVM control data set 29  
 CAVM message data set 29  
 CDBM SUPPORT data set 131  
 created by DFHALTDS job 27  
 created by DFHCOMDS job 27  
 created by DFHDEFDS job 27  
 CSD 63  
 debugging profiles 143  
 creating 143  
 defining 143  
 defining user files 118  
 defining, transient data (extrapartition) 46  
 defining, transient data (intrapartition) 42

data sets *(continued)*  
 DFHAUXT, auxiliary trace 30  
 DFHBUXT, auxiliary trace 30  
 DFHDMPx, dump 30  
 DFHLCD, CICS local catalog 30  
 DFHXRCTL, XRF control data set 105  
 DFHXRMSG, XRF message data set 107  
 DISP option 30  
 dump 99, 188  
 dynamic allocation using CEMT 119  
 for EJB 137  
 for WS-AT 141  
 GTF data sets 28  
 integrity on shared DASD 30  
 messages data set 135  
 MVS system data sets used by CICS 28  
 passively shared 30  
 SDUMP data sets 28  
 SMF data sets 28  
 transient data (extrapartition) 41  
 transient data (intrapartition) 41  
 unique 30  
 user data sets  
 BDAM 116  
 closing 121  
 defining to CICS 118  
 loading VSAM data sets 113  
 opening 120  
 VSAM 112  
 VSAM bases and paths 112  
 XRF considerations 29  
 XRF control data sets 105  
 data tables  
 closing 123  
 loading 123  
 opening 123  
 overview 123  
 types of 123  
 XRF considerations 124  
 database recovery control (DBRC)  
 system initialization parameter, DLDBRC 168  
 use of generic applid 168  
 date format 183  
 DATFORM, system initialization parameter 183  
 DB2 load library  
 requirement for DSNTIAR and DSNTIA1 347  
 DB2 resource security  
 XUSER system initialization parameter  
 AUTHTYPE 263  
 COMAUTHTYPE 263  
 DB2, RCT suffix option of CICS startup 338  
 DB2CONN, system initialization parameter 183  
 DBCTLCON, system initialization parameter 184  
 DBRC (database recovery control)  
 system initialization parameter, DLDBRC 168  
 use of generic applid 168  
 DCT (destination control table)  
 specifying security checking of DCT entries 257  
 DDS option of system initialization parameter  
 BMS 170  
 deadlock timeout 359

- debugging
  - preparing for 363
- debugging profiles data sets
  - creating 143
  - defining 143
    - as remote files 147
    - as VSAM non-RLS files 146
    - as VSAM RLS files 145
- DEBUGTOOL, system initialization parameter 184
- defining debugging profiles data sets
  - as remote files 147
  - as VSAM non-RLS files 146
  - as VSAM RLS files 145
- delay interval, primary, for XRF 220
- delay intervals
  - active delay for XRF 220
  - alternate delay for XRF 164
  - JES for XRF 205
  - reconnection for XRF 168
- delay, persistent verification 226
- destination control table (DCT)
  - specifying security checking of DCT entries 257
- destination request limit, open and close 219
- DEVTYPE macro (MVS) 102
- DFH\$TCTS, copybook 11
- DFH\$TDWT (transient data write-to-terminal sample program) 42
- DFH0STAT, statistics sample program 360
- DFH99, sample DYNALLOC utility program 119
- DFHALTDS, job to create data sets for alternate CICS regions 93
- DFHAUXT auxiliary trace data set 95
- DFHBUXT auxiliary trace data set 95
- DFHCCUTL, local catalog initialization utility program 92
- DFHCMACD, messages data set 27, 136
- DFHCMACI, job to create and initialize the messages data set 27
- DFHCOMDS, job to create common CICS data sets 26
- DFHCOMP1, CSD resource definition group 72
- DFHCOMP2, CSD resource definition group 72
- DFHCOMP3, CSD resource definition group 72
- DFHCSDUP
  - definitions for the Japanese language feature 82
  - moving CICS tables to the CSD 82
- DFHCSDUP offline utility 3
- DFHCSVC, the CICS type 3 SVC 172
- DFHCXRF data set, transient data extrapartition 46
  - DD statements for 47
  - in active CICS regions 46
  - in alternate CICS regions 47
- DFHDBFK data set 132
  - job control statements for CICS execution 132
  - job control statements to define and load 131
- DFHDCTG, group of sample TDQ definitions 41
- DFHDCTG, group of transient data definitions 80
- DFHDEFDS, job to create data sets for each region 26
- DFHDPFMB
  - debugging profiles data set
    - creating 143
- DFHDPFMB (*continued*)
  - defining
    - as remote files 147
    - as VSAM non-RLS files 146
    - as VSAM RLS files 145
- DFHDPFMP
  - debugging profiles data set
    - creating 143
  - defining
    - as remote files 147
    - as VSAM non-RLS files 146
    - as VSAM RLS files 145
- DFHDPFMX
  - debugging profiles data set
    - creating 143
- DFHDU640, dump utility program 101
- DFHDYP, dynamic transaction routing program
  - coding the DTRPGM system initialization parameter 187
  - DFHSIT macro parameters 158
- DFHGCD, global catalog data set 91
- DFHISTAR job 27
- DFHJUP, CICS journal utility program 61
- DFHJVMAT 296
- DFHLCD, local catalog data set 94
- DFHLRQ, BTS data set 351
- DFHNCMN, named counter server region program 411
- DFHNCO macro 407
- DFHNCOPT, named counter options table 407
- DFHRPL, module load library 346
- DFHSIT keywords and operands 157
  - undefined keywords error message 270
- DFHSM (data facility hierarchical storage manager)
  - backup while open 31
- DFHSTART, sample startup procedure 338
- DFHTC2500, close terminal warning message 12
- DFHTC2507, close terminal warning message 12
- DFHTCT5\$, sample TCT 11
- DFHTCTDY, dummy TCT 271
- DFHTEP, terminal error program 13
- DFHTU640, CICS auxiliary trace utility program 25, 98
- DFHXQMN, TS server program 370
- DFHXQMN. system initialization parameters
  - for pool list structure creation 370
- DFHXRCTL, XRF control data set 105
- DFHXRMMSG, XRF message data set 107
- DFLTUSER, system initialization parameter 184
- DIP (data interchange program) 184
- DIP, system initialization parameter 184
- DISMACP, system initialization parameter 185
- DISPLAY command, named counters 416
- DL/I
  - PDIR, system initialization parameter 220
  - specifying security checking of PSB entries 261
- DOCCODEPAGE, system initialization parameter 185
- domains
  - kernel 91
  - parameters 91
- DSA (dynamic storage areas)
  - CDSA 357
  - CICS-key storage 354

DSA (dynamic storage areas) *(continued)*  
   cushions 358  
   CWAKEY, storage key for the CWA 354  
   ECDSA 251, 357  
   ERDSA 354, 357  
   ESDSA 357  
   EUDSA 251, 357  
   key-0 storage 354  
   RDSA 354, 357  
   RENTPGM, storage for read-only DSAs 343  
   RENTPGM, system initialization parameter 227  
   SDSA 357  
   SOS (short-on-storage) 358  
   STGPROT, system initialization parameter 242  
   storage protection facilities 354  
   TCTUAKEY, storage key for terminal control user  
   areas 355  
   UDSA 357  
 DSALIM, system initialization parameter 185  
 DSECT operand of system initialization parameter  
   TYPE 164  
 DSHIPIDL, system initialization parameter 186  
 DSHIPINT, system initialization parameter 186  
 DSNTIA1 347  
 DSNTIAC 347  
 DSNTIAR 347  
 DSRTPGM, system initialization parameter 186  
 DTIMOUT (deadlock timeout interval) 359  
 DTRPGM, system initialization parameter 187  
 DTRTRAN, system initialization parameter 187  
 dump analysis and elimination  
   system initialization parameter 183  
 dump data sets 99, 188  
   dump table facility 99  
   job control statements for CICS execution 103  
   job control statements to allocate 102  
   space calculations 103  
 dump utility program, DFHDU640 101  
 DUMP, system initialization parameter 188  
 DUMPDS, system initialization parameter 188  
 dumps  
   controlling with dump table 99  
   effect of START= parameter 284  
 DUMPSW, system initialization parameter 188  
 DURETRY, system initialization parameter 188  
 dynamic allocation  
   ADYN, dynamic allocation transaction 119  
   DFH99, sample DYNALLOC utility program 119  
 dynamic allocation of the CSD 81  
 dynamic transaction routing program, DFHDYP  
   coding the DTRPGM system initialization  
   parameter 187  
   DFHSIT macro parameters 158

## E

ECDSA (extended CICS-key DSA) 357  
 ECDSASZE, system initialization parameter 189  
 EDSALIM, system initialization parameter 189  
 EJBROLEPRFX, system initialization parameter 190

emergency restart  
   resource backout 247  
   START system initialization parameter 239  
 ENCRYPTION, system initialization parameter 191  
 EODI, system initialization parameter 194  
 ERDSA (extended read-only DSA) 354, 357  
 ERDSASZE, system initialization parameter 194  
 ESDSA (shared DSA) 357  
 ESDSASZE, system initialization parameter 194  
 ESMEXITS, system initialization parameter 195  
 EUDSA (extended user DSA) 357  
 EUDSASZE, system initialization parameter 195  
 exception class monitoring 212  
 EXEC CICS CREATE commands 3  
 exit interval, region 203  
 exits  
   FE, global trap exit 251  
   IEFUSI, exit routine 353  
   XDUCLSE, dump global user exit 102  
   XDUOUT, dump global user exit 102  
   XDUREQ, dump global user exit 102  
   XDUREQC, dump global user exit 102  
 extended recovery facility (XRF)  
   terminal considerations 20  
 external security interface 232  
 extrapartition transient data 41  
   CSSL, and other destinations used by CICS 348  
 extrapartition transient data queues 348

## F

facilities  
   autoinstall 4  
   auxiliary trace autoswitch facility 169  
   generalized trace facility (GTF) 28  
   storage protection facilities 354  
   temporary storage 35  
 FCT (file control table)  
   specifying the FCT suffix 195  
 FCT, system initialization parameter 195  
 FE global trap exit 251  
 FEPI (front end programming interface)  
   CSZL, transient data queue 42  
   CSZX, transient data queue 42  
   FEPI, system initialization parameter 196  
 field name start character 196  
 field separator characters 196  
 file control table (FCT)  
   specifying the FCT suffix 195  
 FILEA 27  
 FILSTAT operand 123  
 FLDSEP, system initialization parameter 196  
 FLDSTRT, system initialization parameter 196  
 FORCEQR, system initialization parameter 197  
 frequency, activity keypoint 167  
 front end programming interface (FEPI)  
   CSZL, transient data queue 42  
   CSZX, transient data queue 42  
   FEPI, system initialization parameter 196  
 FSSTAFF, system initialization parameter 197

FULL option of system initialization parameter  
BMS 169

## G

GCD (global catalog data set)  
buffer space 87  
description 85  
job control statement for CICS execution 91  
job control statements to define and initialize 86  
space calculations 89  
generalized trace facility (GTF) 28  
generic applid for CICS XRF regions 167  
global catalog  
AMP parameter for specification 348  
for resource definitions 278  
use in restart 278  
global catalog data set (GCD)  
buffer space 87  
description 85  
job control statement for CICS execution 91  
job control statements to define and initialize 86  
space calculations 89  
global trap exit, FE 251  
GMTEXT, system initialization parameter 198  
GMTRAN, system initialization parameter 199  
GNTRAN, system initialization parameter 199  
good morning message 198  
good morning transaction 199  
GOOD MORNING transaction 229  
group list, RDO 201  
GRPLIST, system initialization parameter 201  
GTF (generalized trace facility) 28  
GTFTR, system initialization parameter 202

## H

high-performance option (HPO) 203  
HPO (high-performance option) 203  
HPO, system initialization parameter 203

## I

I/O error handling 110  
ICP (interval control program) 203  
ICP, system initialization parameter 203  
ICV, system initialization parameter 203  
ICVR, system initialization parameter 204  
ICVTSD, system initialization parameter 204  
IDCAMS, AMS utility program 113  
IEFUSI, exit routine 353  
IEV017 error message 270  
IIOPLISTENER, system initialization parameter 204  
INFOCENTER, system initialization parameter 204  
INITIAL  
system initialization parameter START 240  
initial start  
START system initialization parameter 239  
INITPARM, system initialization parameter 204  
interactive problem control system (IPCS) 101  
internal trace, main storage 205

interregion communication (IRC) 205  
intersystem communication (ISC) 205  
interval control program (ICP) 203  
intervals, activity keypoint 167  
intrapartition transient data 41, 347  
intrapartition transient data queues  
defining the intrapartition data set 42  
INTTR, system initialization parameter 205  
IPCS (interactive problem control system) 101  
IRC (interregion communication) 205  
IRCSTRT, system initialization parameter 205  
ISC (intersystem communication) 205  
ISC, system initialization parameter 205

## J

Japanese language feature  
installing definitions in the CSD 82  
Java  
system properties 296  
JCL (job control language)  
CICS startup 340  
as a batch job 340  
as a started task 360  
JES delay interval for XRF 205  
JESDI, system initialization parameter 205  
job control language (JCL)  
for CICS as a batch job 340  
for CICS as a started task 360  
job streams  
CICS startup 338  
defining DFHXRMMSG data set 107  
defining XRF control data set 106  
jobs  
DFHALTDS, job to create data sets for alternate  
CICS regions 93  
DFHCMACI, job to create and initialize the messages  
data set 27  
DFHCOMDS, job to create common CICS data  
sets 26  
DFHDEFDS, job to create data sets for each  
region 26, 35, 145  
DFHISTAR 27  
journaling  
BWO 31  
specifying security checking for journal entries 259  
XJCT, system initialization parameter 259  
JOURNALMODEL definitions 55  
JVMCCPROFILE, system initialization parameter 205  
JVMCCSIZE, system initialization parameter 206  
JVMCCSTART, system initialization parameter 206  
JVMLEVEL0TRACE, system initialization  
parameter 206  
JVMLEVEL1TRACE, system initialization  
parameter 206  
JVMLEVEL2TRACE, system initialization  
parameter 206  
JVMPROFILEDIR, system initialization parameter 207  
JVMUSERTRACE, system initialization parameter 206

## K

key-0 storage 354  
keypoint frequency 167  
KEYRING, system initialization parameter 208  
keys for page-retrieval 234

## L

Language Environment run-time library,  
SCEERUN 340  
Language Environment run-time library,  
SCEERUN2 340  
LCD (local catalog data set)  
description 91  
job control statement for CICS execution 94  
job control statements to define and initialize 93  
use in restart 279  
LGDFINT system initialization parameter 208  
LGNMSG, system initialization parameter 209  
libraries  
SCEERUN, Language Environment run-time  
library 340  
SCEERUN2, Language Environment run-time  
library 340  
line printer 10  
LLACOPY macro 209  
LLACOPY, system initialization parameter 209  
load modules  
DFHFCT, FCT load module 5  
local catalog data set (LCD)  
description 91  
job control statement for CICS execution 94  
job control statements to define and initialize 93  
use in restart 279  
LOCALCCSID, system initialization parameter 209  
locating modules in the relocatable program  
library 209  
log defer time interval 208  
log manager 49  
log streams  
mapping system log and journal names to 56  
LOGA transient data destination 348  
logging  
defining CICS journals for general logs  
forward recovery logs 53  
user journals 54  
defining CICS logs 49  
defining CICS system logs 49  
defining dummy logs 50  
JOURNALMODEL definitions 55  
log autoinstall 55  
logon data, VTAM 209  
LPA (link pack area)  
LPA system initialization parameter 209  
PRVMOD system initialization parameter 224  
LPA, system initialization parameter 209

## M

macro definition 3

## macros

ASLTAB (VTAM macro) 8  
DEVTYPE (MVS macro) 102  
macro instructions 5  
MDLTAB (VTAM macro) 8  
MGCR (to issue MVS commands) 17  
MVS SDUMP 28  
MAXJVMTCBS, system initialization parameter 152  
MAXOPENTCBS, system initialization parameter 152  
MAXSOCKETS, system initialization parameter 210  
MAXXPTCBS, system initialization parameter 152  
MCT (monitoring control table) 211  
MCT, system initialization parameter 211  
MDLTAB (VTAM macro) 8  
message case 215  
message level 215  
message, good morning 198  
messages  
crucial and non-crucial messages 109  
DFHXRMMSG, XRF message data set 107  
messages data set  
job control statements for CICS execution 136  
job control statements to define and load 135  
messages data set for CMAC facility 27, 135  
methods of resource definition 4  
MGCR macro, to issue MVS commands 17  
MINIMUM option of system initialization parameter  
BMS 169  
MN, system initialization parameter 211  
MNCONV, system initialization parameter 212  
MNEXC, system initialization parameter 212  
MNFREQ, system initialization parameter 212  
MNPER, system initialization parameter 213  
MNRES, system initialization parameter 213  
MNSUBSYS, system initialization parameter 213  
MNSYNC, system initialization parameter 213  
MNTIME, system initialization parameter 213  
MODIFY command 14  
module load library concatenation, DFHRPL 346  
monitoring 211, 283  
exception class 212  
performance class 213  
transaction resource monitoring 213  
monitoring control table (MCT) 211  
MRO (multiregion operation)  
batching 214  
batching requests 214  
extend lifetime of long-running mirror 214  
long-running mirror 215  
MROBTCH, system initialization parameter 214  
MROFSE, system initialization parameter 214  
MROLRM, system initialization parameter 215  
MSGCASE, system initialization parameter 215  
MSGLVL, system initialization parameter 215  
multiregion operation (MRO)  
batching 214  
batching requests 214  
extend lifetime of long-running mirror 214  
long-running mirror 215  
MVS SDUMP macro 99  
MVS START command, to start CICS 360

MXT, system initialization parameter 215

## N

named counters

- automatic restart manager parameters 413
- controlling server regions 415
- debug trace parameters 414
- defining an options table 407
- defining and starting server 411
- deleting or emptying pools 418
- DISPLAY command 416
- dumping pool list structures 421
- list structure parameters 414
- list structure, defining 409
- options table
  - defining 407
  - making available to CICS 409
  - parameters 407
- parameters, server 412
- PRINT command 416
- security 407
- server overview 405
- server parameters 412
- SET command 416
- starting a server 405
- structures and servers 406
- unloading and reloading pools 419
- warning parameters 415
- XES 418

NATLANG, system initialization parameter 216

NCPLDFT, system initialization parameter 218

NEWSIT, system initialization parameter 218

effect on warm start 280

NODDS option of system initialization parameter

BMS 170

non-VTAM terminals 5

## O

open destination request limit 219

open transaction environment 152

open transaction environment (OTE) TCBS 152

operator communication for initialization  
parameters 278

OPERTIM, system initialization parameter 219

OPNDLIM, system initialization parameter 219

OPNDST write-to-operator timeout limit 219

OS/390 console, defining to CICS 14

OTE TCBS 152

overriding system initialization parameters

from the console 277

from the SYSIN data set 276

overview of coupling facility data table server 383

overview of temporary storage data sharing server 369

## P

PA keys for page-retrieval 234

PA keys for screen copying 223

page-chaining command character string 221

page-copying command character string 221

page-purging command character string 221

page-retrieval command character string 221

page-retrieval keys 234

PARM startup parameter

system initialization parameters 343

PARMERR, system initialization parameter 220

PDI, system initialization parameter 220

PDIR, system initialization parameter 220

performance class monitoring 213

persistent sessions, VTAM 18

persistent verification delay 226

PF keys for page-retrieval 234

PGCHAIN, system initialization parameter 221

PGCOPY, system initialization parameter 221

PGPURGE, system initialization parameter 221

PGRET, system initialization parameter 221

PL/I language support

transient data destinations 348

PLT (program list table)

system initialization programs 222

system termination programs 223

PLTPI, system initialization parameter 222

PLTPISEC, system initialization parameter 222

PLTPIUSR, system initialization parameter 222

PLTSD, system initialization parameter 223

PRGDLAY, system initialization parameter 223

primary delay interval for XRF 220

PRINT command, named counters 416

PRINT, system initialization parameter 223

program list table (PLT) 338

system initialization programs 222

system termination programs 223

program specification block (PSB)

PDIR, system initialization parameter 220

specifying security checking of PSB entries 261

PRTYAGE, system initialization parameter 224

PRVMOD, system initialization parameter 224

PSB (program specification block)

PDIR, system initialization parameter 220

specifying security checking of PSB entries 261

PSBCHK, system initialization parameter 225

PSDINT, system initialization parameter 159, 225

PSTYPE, system initialization parameter 159, 226

purge delay time interval, BMS 223

PVDELAY, system initialization parameter 226

## R

RACF (resource access control facility)

checking program entries with RACF 260

determining results of RACF authorization

requests 233

DFLTUSER, system initialization parameter 184

establishing APPC sessions 233

MRO bind-time security 233

protecting your data sets 23, 30

resource level checking 233

SEC, system initialization parameter 232

SECPRFX, system initialization parameter 234

specifying a prefix to resource name 234

RACF (resource access control facility) *(continued)*  
 XAPPC, system initialization parameter 256  
 RAMAX, system initialization parameter 226  
 RAPOOL, system initialization parameter 226  
 RBA (relative byte address) 44  
 RDO (resource definition online)  
 CICS system definition data set (CSD) 63  
 command logs  
 CADL 64, 80  
 CAIL 64, 80  
 CRDI 64, 80  
 CSDL 64, 80  
 CSFL 64, 80  
 CSKL 64, 80  
 CSPL 64, 80  
 CSRL 64, 80  
 group list (GRPLIST) 201  
 RDSA (read-only DSA) 354, 357  
 RDSASZE, system initialization parameter 227  
 read-only storage  
 system initialization parameter 227  
 RECEIVE ANY (RA) maximum 226  
 RECEIVE ANY (RA) pool size 226  
 reconnection delay interval (XRF) 168  
 reconnection transaction for XRF 229  
 record-level sharing (RLS)  
 VSAM data sharing 114  
 region exit interval (ICV) 203  
 REGION parameter for CICS startup 352  
 relative byte address (RBA) 44  
 RELOAD, named counter pool 419  
 RENTPGM, storage for read-only DSAs 343  
 RENTPGM, system initialization parameter 227  
 request parameter list (RPL) 226  
 resource access control facility (RACF)  
 checking program entries with RACF 260  
 determining results of RACF authorization requests 233  
 DFLTUSER, system initialization parameter 184  
 establishing APPC sessions 233  
 MRO bind-time security 233  
 protecting your data sets 23, 30  
 resource level checking 233  
 SEC, system initialization parameter 232  
 SECPRFX, system initialization parameter 234  
 specifying a prefix to resource name 234  
 XAPPC, system initialization parameter 256  
 resource backout at emergency restart 247  
 resource definition  
 methods of 4  
 resource definition online (RDO) 3  
 CICS system definition data set (CSD) 63  
 command logs  
 CADL 64, 80  
 CAIL 64, 80  
 CRDI 64, 80  
 CSDL 64, 80  
 CSFL 64, 80  
 CSKL 64, 80  
 CSPL 64, 80  
 CSRL 64, 80

resource definition online (RDO) *(continued)*  
 group list (GRPLIST) 201  
 resources  
 ways to define 4  
 RESP, system initialization parameter 228  
 RESSEC, system initialization parameter 228  
 RLS, record-level sharing  
 VSAM data sharing 114  
 RMTRAN, system initialization parameter 229  
 RPL (request parameter list) 226  
 RRMS, system initialization parameter 230  
 RST, system initialization parameter 230  
 RSTSIGNOFF, system initialization parameter 230  
 RSTSIGTIME, system initialization parameter 231  
 RUWAPOOL, system initialization parameter 231

## S

sample job streams  
 CICS startup 338  
 defining DFHXRMMSG data set 107  
 defining XRF control data set 106  
 sample program file, FILEA 27  
 samples  
 data for loading a BDAM data set 117  
 DFH\$TDWT (transient data write-to-terminal program) 42  
 DFHDCTG, queue definitions 41  
 DFHSTART, sample startup procedure 338  
 DFHTCT5\$, sample TCT 11  
 FILEA sample program file 27  
 JCL to create and load a BDAM data set 116  
 job stream  
 CICS startup 338  
 to define DFHXRMMSG data set 107  
 to define XRF control data set 106  
 sample job to define auxiliary data sets on disk 96  
 screen copying 223  
 SDSA (shared DSA) 357  
 SDSASZE, system initialization parameter 232  
 SDTRAN, system initialization parameter 232  
 SDUMP data sets 28  
 SDUMP macro 99  
 CICS retry interval 188  
 DURETRY option 188  
 SEC, system initialization parameter 232  
 SECPRFX, system initialization parameter 234  
 security  
 for transactions 262  
 MRO bind-time security 233  
 of attached entries 262  
 of XRF data sets 110  
 protecting your data sets 23, 30  
 resource class names 256  
 SEC, system initialization parameter 232  
 SECPRFX, system initialization parameter 234  
 security checking  
 for EXEC CICS system commands 256  
 for program entries 260  
 for temporary storage entries 262  
 of DB2 resources 257

security (*continued*)

- security checking (*continued*)
  - of destination control entries 257
  - of enterprise bean method calls 258
  - of EXEC-started transaction entries 259
  - of file control entries 258
  - of journal entries 259
  - of PSB entries 261
- specifying a prefix to resource name 234
- using RACF to establish APPC sessions 233
- XAPPC, system initialization parameter 256
- XCMD, system initialization parameter 256
- XDB2, system initialization parameter 257
- XDCT, system initialization parameter 257
- XEJB, system initialization parameter 258
- XFCT, system initialization parameter 258
- XJCT, system initialization parameter 259
- XPCT, system initialization parameter 259
- XPPT, system initialization parameter 260
- XPSB, system initialization parameter 261
- XTRAN, system initialization parameter 262
- XTST, system initialization parameter 262

sequence numbers

- See named counters 405

sequential devices 12

sequential terminal devices 10

- closing (quiescing) the devices 12
- coding input for 12
- DFHTC2500, close terminal warning message 12
- DFHTC2507, close terminal warning message 12
- end-of-file 12
- logical close to quiesce 352
- terminating input data 12

SET command, named counters 416

SETXCF, delete named counter pool 418

sharing the CSD 68

- freeing internal locks 73
- protection by internal locks 69
- sharing between CICS regions 70

single keystroke retrieval (SKR) 234

SIT (system initialization table)

- default SIT (DFHSIT) 263
- DFHSIT keywords and operands 157
- DFHSIT TYPE=CSECT 164
- DFHSIT TYPE=DSECT 164
- supplying system initialization parameters to CICS 273

SIT, system initialization parameter 234

SKR (single keystroke retrieval) 234

SKRxxxx, system initialization parameter 234

SMS (storage management subsystem) 31, 33, 34

SNSCOPE, system initialization parameter 235

SOS (short-on-storage) 358

space calculations

- auxiliary trace data set 97
- CAVM 106
- CSD 64
- defining data sets 23
- disk space 64
- dump data sets 103
- global catalog 89
- space calculations (*continued*)
  - XRF message data set 108
- SPCTR, system initialization parameter 235
- SPCTRxx, system initialization parameter 236
- SPOOL, system initialization parameter 238
- SRBSVC, system initialization parameter 238
- SRT (system recovery table) 238, 239
- SRT, system initialization parameter 238
- SRVERCPsystem initialization parameter 238
- SSLDELAY, system initialization parameter 239
- SSLTCB, system initialization parameter 239
- STANDARD option of system initialization parameter
  - BMS 169
- STANDBY start option 240
- standby start-up for XRF 240
- START command, MVS 360
- START, system initialization parameter 239
  - (option,ALL) 240
  - START=AUTO 279
  - START=COLD 281
  - START=INITIAL 280
  - START=STANDBY 281
- started task, CICS as a 360
- STARTER, system initialization parameter 241
- starting CICS regions 337
  - as a started task 360
  - MVS START command 360
  - sample job stream 338
  - specifying the type of startup 278
  - START=AUTO 279
  - START=COLD 281
  - START=INITIAL 280
  - START=STANDBY, for an XRF alternate CICS 281
- startup job streams
  - terminals 7
- startup procedure, DFHSTART 338
- STATEOD, system initialization parameter 241
- STATINT, system initialization parameter 241
- statistics 283
- statistics sample program, DFH0STAT 360
- STATRCD, system initialization parameter 241
- STGPROT, system initialization parameter 242
- STGRCVY, system initialization parameter 242
- STNTR, system initialization parameter 243
- STNTRxx, system initialization parameter 243
- storage calculations 38
- storage calculations in brief 38
- storage cushion size
  - short-on-storage warnings 359
- storage management subsystem (SMS) 31, 33, 34
  - backup while open facility 31
  - introduction 34
  - release information 33
- storage protection for CICS regions 354
- storage protection system initialization parameter,
  - STGPROT 242
- storage requirements for CICS regions 352
- storage trace
  - auxiliary 169
  - main 205
  - trace option in transaction dump 252

storage trace (*continued*)  
 trace table size in main storage 251  
 trace table size in transaction dump 252  
 strings and buffers, VSAM 249, 252  
 SUBTSKS, system initialization parameter 245  
 SUFFIX, system initialization parameter 245  
 supervisor call (SVC)  
 type 3, DFHCSVC 172  
 type 6, DFHPSVC 203, 238  
 surveillance signal for XRF 105, 164  
 SVC (supervisor call)  
 type 3, DFHCSVC 172  
 type 6, DFHPSVC 203, 238  
 SYSIDNT, system initialization parameter 245  
 SYSIN data stream 344  
 SYSIN, control keyword 273  
 system console 14  
 system data sets 23  
 system dumps 99  
 system identifier, system initialization parameter  
 SYSIDNT 245  
 system initialization  
 for an alternate CICS (XRF=YES)  
 START=STANDBY 281  
 how CICS determines the type of startup 278  
 START=AUTO 279  
 START=COLD 281  
 START=INITIAL 280  
 system initialization parameter  
 EJBROLEPRFX 190  
 ENCRYPTION 191  
 system initialization parameters  
 ADI 164  
 AIBRIDGE 164  
 AICONS 164  
 AIEXIT 165  
 AILDELAY 165  
 AIQMAX 166  
 AIRDELAY 166  
 AKPFREQ 167  
 APPLID 167  
 AUTCONN 168  
 AUTODST 168  
 AUTORESETTIME 168  
 AUXTR 169  
 AUXTRSW 169  
 BMS 169  
 BRMAXKEEPTIME 171  
 CDSASZE 171  
 CHKSTRM 171  
 CHKSTSK 171  
 CICSSVC 172  
 CILOCK 172  
 CLINTCP 172  
 CLSDSTP 172  
 CLT 173  
 CMDPROT 173  
 CMDSEC 174  
 CONFDATA 174  
 CONFTXT 176  
 CPSMCONN 176

system initialization parameters (*continued*)  
 CRLPROFILE 177  
 CSDACC 177  
 CSDBKUP 178  
 CSDBUFND 178  
 CSDBUFNI 178  
 CSDDISP 178  
 CSDDSN 179  
 CSDFRLOG 179  
 CSDJID 180  
 CSDLRNO 181  
 CSDRECOV 181  
 CSDSTRNO 182  
 CWAKEY 182  
 DAE 183  
 DATFORM 183  
 DB2CONN 183  
 DBCTLCON 184  
 DEBUGTOOL 184  
 DFLTUSER 184  
 DIP 184  
 DISMACP 185  
 DOCCODEPAGE 185  
 DSALIM (DSA storage limit) 185  
 DSHIPIDL 186  
 DSHIPINT 186  
 DSRTPGM 186  
 DTRPGM 187  
 DTRTRAN 187  
 DUMP 188  
 DUMPDS 188  
 DUMPSW 188  
 DURETRY 188  
 ECDSASZE 189  
 EDSALIM (EDSA storage limit) 189  
 entering at the console 277  
 EODI 194  
 ERDSASZE 194  
 ESDSASZE 194  
 ESMEXITS 195  
 EUDSASZE 195  
 FCT 195  
 FEPI 196  
 FLDSEP 196  
 FLDSTRT 196  
 FORCEQR 197  
 from operator's console 273, 278  
 from the SYSIN data set 344  
 FSSTAFF 197  
 GMTEXT 198  
 GMTRAN 199  
 GNTRAN 199  
 GRPLIST 201  
 GTFTR 202  
 how to specify 151  
 HPO 203  
 ICP 203  
 ICV 203  
 ICVTSD 204  
 IIOPLISTENER 204  
 in the PARM parameter 273, 276

system initialization parameters *(continued)*

in the SYSIN data set 273, 276

INFOCENTER 204  
 INITPARM 204  
 INTTR 205  
 IRCSTRT 205  
 ISC 205  
 JESDI 205  
 JVMCCPROFILE 205  
 JVMCCSIZE 206  
 JVMCCSTART 206  
 JVMLEVELOTRACE 206  
 JVMLEVEL1TRACE 206  
 JVMLEVEL2TRACE 206  
 JVMPROFILEDIR 207  
 JVMUSERTRACE 206  
 KEYRING 208  
 LGDFINT 208  
 LGNMSG 209  
 LLACOPY 209  
 LOCALCCSID 209  
 LPA 209  
 MAXJVMTCBS 152  
 MAXOPENTCBS 152  
 MAXSOCKETS 210  
 MAXXPTCBS 152  
 MCT 211  
 MN 211  
 MNCONV 212  
 MNEXC 212  
 MNFREQ 212  
 MNPER 213  
 MNRES 213  
 MNSUBSYS 213  
 MNSYNC 213  
 MNTIME 213  
 MROBTCH 214  
 MROFSE 214  
 MROLRM 215  
 MSGCASE 215  
 MSGLVL 215  
 NATLANG 216  
 NCPLDFT 218  
 NEWSIT 218  
 OPERTIM 219  
 OPNDLIM 219  
 PARMERR 220  
 PDI 220  
 PDIR 220  
 PGAICTLG 220  
 PGAEXIT 221  
 PGAIPGM 221  
 PGCHAIN 221  
 PGCOPY 221  
 PGPURGE 221  
 PGRET 221  
 PLTPI 222  
 PLTPISEC 222  
 PLTPIUSR 222  
 PLTSD 223  
 PRGDLAY 223

system initialization parameters *(continued)*

PRINT 223  
 PRTYAGE 224  
 PRVMOD 224  
 PSBCHK 225  
 PSDINT 225  
 PSTYPE 226  
 PVDELAY 226  
 RAMAX 226  
 RAPOOL 226  
 RDSASZE 227  
 RENTPGM 227  
 RESP 228  
 RESSEC 228  
 RMTRAN 229  
 RRMS 230  
 RST 230  
 RSTSIGNOFF 230  
 RSTSIGTIME 231  
 RUWAPOOL 231  
 SDSASZE 232  
 SDTRAN 232  
 SEC 232  
 SECPRFX 234  
 SIT 234  
 SKRxxxx 234  
 SNSCOPE 235  
 SPCTR 235  
 SPCTRTS 236  
 SPCTRxx 236  
 specifying on the PARM statement 343  
 SPOOL 238  
 SRBSVC 238  
 SRT 238  
 SRVERCP 238  
 SSLDELAY 239  
 SSLTCB 239  
 START 239  
 STARTER 241  
 STATEOD 241  
 STATINT 241  
 STATRCD 241  
 STGPROT 242  
 STGRCVY 242  
 STNTR 243  
 STNTRxx 243  
 SUBTSKS 245  
 SUFFIX 245  
 SYSIDNT 245  
 SYSTR 245  
 TAKEOVR 246  
 TBEXITS 247  
 TCAM 247  
 TCP 247  
 TCPIP 247  
 TCSACTN 247  
 TCSWAIT 248  
 TCT 249  
 TCTUAKEY 249  
 TCTUALOC 249  
 TD 249

system initialization parameters (*continued*)

TDINTRA 250  
TDSUBTASK 250  
TRANISO (transaction isolation) 250  
TRAP 251  
TRTABSZ 251  
TRTRANSZ 252  
TRTRANTY 252  
TS 252  
TST 252  
TYPE 164  
UDSASZE 253  
UOWNETQL 253  
USERTR 253  
USRDELAY 253  
VTAM 254  
VTPREFIX 255  
WEBDELAY 256  
WRKAREA 256  
XAPPC 256  
XCMD 256  
XDB2 257  
XDCT 257  
XEJB 258  
XFCT 258  
XJCT 259  
XLT 259  
XPCT 259  
XPPT 260  
XPSB 261  
XRF 261  
XRFSOFF 261  
XRFSTME 262  
XTRAN 262  
XTST 262  
XUSER 263

System initialization parameters  
ICVR 204  
MXT 215

system initialization table (SIT)  
default SIT (DFHSIT) 263  
DFHSIT keywords and operands 157  
DFHSIT TYPE=CSECT 164  
DFHSIT TYPE=DSECT 164  
supplying system initialization parameters to CICS 273

system management facilities  
for CICS statistics 28

system programming  
EXEC CICS CREATE commands 3

system recovery table (SRT) 238, 239

system spooling interface 238

system startup 337  
startup job stream 338

system task  
See started task, CICS as a

System-managed duplexing 433

System-managed rebuild 431

SYSTR, system initialization parameter 245

## T

takeover action for XRF 246

TAKEOVR, system initialization parameter 246

TBEXITS, system initialization parameter 247

TCAM, system initialization parameter 247

TCP, system initialization parameter 247

TCPIP, system initialization parameter 247

TCSACTN, system initialization parameter 247

TCSWAIT, system initialization parameter 248

TCT (terminal control table) 249

TCT, system initialization parameter 249

TCTUAKEY, storage key for terminal control user areas 355

TCTUAKEY, system initialization parameter 249

TCTUALLOC, system initialization parameter 249

TD, system initialization parameter 249

TDINTRA, system initialization parameter 250

TDSUBTASK, system initialization parameter 250

temporary storage  
starting up temporary storage servers 369  
TS data sharing server 369  
VSAM buffers and strings 252

temporary storage data sharing  
defining TS pools for TS data sharing 38

temporary storage server  
sample startup job 371

temporary storage table (TST) 252  
specifying security checking of temporary storage entries 262

terminal control table (TCT) 249  
dummy control table, DFHTCTDY 271

terminal control table user area storage key  
system initialization parameter 249

terminal definition 7

terminal error program, DFHTEP 13

terminal scan delay, ICVTSD 204

terminals, defining 7

terminating 12

time interval, region exit 203

timeout limit, userid 253

trace  
auxiliary storage trace 169  
auxiliary trace autoswitch facility 169  
auxiliary trace data sets for XRF 97  
AUXTR, system initialization parameter 169  
AUXTRSW, system initialization parameter 169  
CETR, trace control transaction 96  
CICS standard tracing, setting levels of 243  
controlling trace with CEMT or CETR 96  
defining auxiliary trace data sets 95  
DFHAUXT auxiliary trace data set 95  
DFHBUXT auxiliary trace data set 95  
DFHTU640, trace utility program 98  
GTFTR, system initialization parameter 96, 202  
INTTR, system initialization parameter 96, 205  
job control statements to allocate auxiliary trace data sets 96  
option in transaction dump 252  
sample job to define auxiliary data sets on disk 96  
SM component, warning when setting trace level 243

trace (*continued*)

- space calculations for auxiliary trace data sets 97
- SPCTR, system initialization parameter 236
- SPCTRxx, system initialization parameter 236
- special tracing, setting levels of 236
- starting auxiliary trace 95
- STNTR, system initialization parameter 243
- STNTRxx, system initialization parameter 243
- SYSTR, system initialization parameter 245
- table size in main storage 251
- table size in transaction dump 252
- TRTABSZ, system initialization parameter 251
- TRTRANSZ, system initialization parameter 252
- TRTRANTY, system initialization parameter 252
- USERTR, system initialization parameter 96, 253
- using auxiliary trace data sets 95
- using DFHDEFDS to allocate auxiliary trace data sets 96

trace utility program, DFHTU640 98

TRANISO, system initialization parameter 250

transaction isolation 356

transaction list table, XLT 259

transaction resource monitoring 213

transactions

- ADYN, dynamic allocation transaction 119
- CEDA 5
- CEDB 5
- CEDC 5
- CESF GOODNIGHT 12
- CESF LOGOFF 13
- CESN 16
- CSFU, CICS file utility transaction 120

transient data (extrapartition) data sets 41

- defining 46

transient data (intrapartition) data set

- defining 42, 43
- failure to open 43
- job control statements to define 43
- multiple extents and volumes 43
- other considerations 44
- VSAM data set 44
  - control interval size 44
  - job control statement for CICS execution 45
  - space considerations 44
- XRF considerations 45

transient data queues 41

transient data write-to-terminal sample program (DFH\$TDWT) 42

TRAP, system initialization parameter 251

TRTABSZ, system initialization parameter 251

TRTRANSZ, system initialization parameter 252

TRTRANTY, system initialization parameter 252

TS, system initialization parameter 252

TSO users 16

TST (temporary storage table) 252

- specifying security checking of temporary storage entries 262

TST, system initialization parameter 252

TYPE, system initialization parameter 164

TYPE=CSECT, DFHSIT 164

TYPE=DSECT, DFHSIT 164

types of data tables 123

## U

UDSA (user DSA) 357

UDSASZE, system initialization parameter 253

UNLOAD, named counter pool 419

UOWNETQL, system initialization parameter 253

user file definitions 111

user files

- coupling facility data table server 383

userid timeout limit 253

USERTR, system initialization parameter 253

USRDELAY, system initialization parameter 253

utility programs

- DFHCCUTL, local catalog initialization utility program 92
- DFHDU640, dump utility program 101
- DFHJUP, CICS journal utility program 61
- DFHTU640, CICS auxiliary trace utility program 25
- IDCAMS, AMS utility program 113

## V

VERBEXIT, IPCS parameter 101

virtual telecommunications access method (VTAM)

- ACB at CICS startup 284
- high performance option (HPO) 203
- logon data 208, 209
- system initialization parameter, VTAM 254
- terminals, statements for 7
- VBUILD TYPE=APPL statement 167

virtual terminals

- VTPREFIX 255

VSAM buffers and strings 249, 252

VSAM data sets 112

- bases and paths 112
- loading empty VSAM data sets 113
- opening and closing 120

VSAM intrapartition data set 41

VTAM

- persistent sessions 18

VTAM, system initialization parameter 254

VTPREFIX, system initialization parameter 255

## W

warm start 239

WEBDELAY, system initialization parameter 256

welcome (good morning) message 198

write-to-operator timeout limit 219

WRKAREA, system initialization parameter 256

## X

XAPPC, system initialization parameter 256

XCMD, system initialization parameter 256

XDB2, system initialization parameter 257

XDCT, system initialization parameter 257

XDUCLSE, dump global user exit 102

XDUOUT, dump global user exit 102  
 XDUREQ, dump global user exit 102  
 XDUREQC, dump global user exit 102  
 XEJB, system initialization parameter 258  
 XES, named counter server response to 418  
 XFCT, system initialization parameter 258  
 XJCT, system initialization parameter 259  
 XLT, system initialization parameter 259  
 XLT, transaction list table 259  
 XPCT, system initialization parameter 259  
 XPPT, system initialization parameter 260  
 XPSB, system initialization parameter 261  
 XRF  
     terminal considerations 20  
 XRF (extended recovery facility)  
     actively shared data sets 29  
     ADI (alternate) 164  
     AIRDELAY parameter (active and alternate CICS) 166  
     allocation and dispositions of data sets 29  
     alternate delay 164  
     alternate delay interval 164  
     APPLID system initialization parameter 167  
     AUTCONN, system initialization parameter 168  
     auxiliary trace data sets 97  
     CLT system initialization parameter 173  
     command list table (CLT) 173  
     control data sets 105  
     CSD requirements 82  
     data set considerations when running CICS with XRF 29  
     data set status 29  
     DD statements in CICS startup job (DFHXRCTL) 107  
     DFHCXRF data set for the alternate CICS 47  
     DFHXRMSG, message data set 107  
     DISP option for data sets 30  
     DISP=SHR 82  
     DUMP system initialization parameter 188  
     generic and specific applids 167  
     GOOD MORNING transaction 229  
     integrity of data on shared DASD 30  
     JCL to define XRF message data set 107  
     JES delay interval 205  
     JESDI system initialization parameter 205  
     job stream to define DFHXRCTL 106  
     passively shared data sets 30  
     PDI system initialization parameter 220, 229  
     primary delay interval (PDI) 220  
     reconnection delay 168  
     reconnection transaction 229  
     space calculations for DFHXRMSG 108  
     START=STANDBY (alternate) 240  
     surveillance signal 164  
     TAKEOVR system initialization parameter 246  
     temporary storage data set 37  
     transient data extrapartition data set 47  
     transient data intrapartition data set 45  
     user file definitions 121  
     VTAM ACB at startup 284  
     XRF system initialization parameter 261  
     XRF, system initialization parameter 261  
     XRFSOFF, system initialization parameter 261  
     XRFSTME, system initialization parameter 262  
     XTRAN, system initialization parameter 262  
     XTST, system initialization parameter 262  
     XUSER, system initialization parameter 263



---

## 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](http://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](http://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.







Product Number: 5655-M15

SC34-6428-08



Spine information:



CICS TS for z/OS

CICS System Definition Guide

Version 3  
Release 1