CICS® Transaction Server for OS/390®

CICS System Definition Guide

Release 3

SC33-1682-02

CICS® Transaction Server for OS/390®

CICS System Definition Guide

Release 3

SC33-1682-02

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

Third edition (March 1999)

This edition applies to Release 3 of CICS Transaction Server for OS/390, program number 5655-147, 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.

This edition replaces and makes obsolete the previous edition, SC33-1682-01. The technical changes for this edition are summarized under ″Summary of changes″ and are indicated by a vertical bar to the left of a change.

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

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

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

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

ACF/VTAM	AD/Cycle
BookManager	CICS
CICS/ESA	CICS/MVS
CICS/VSE	COBOL/370
DATABASE 2	DB2
DFSMS	ESA/390
Hiperbatch	IBM
IBMLink	IMS/ESA
Language Environment	MVS/DFP
MVS/ESA	MVS/SP
MVS/XA	NetView
OS/390	Processor Resource/Systems Manager
PR/SM	RACF
SAA	Systems Application Architecture
VSE/ESA	VTAM

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

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 91 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 OS/390.

“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

Part 1. Installing resource definitions ... pages 1—90
Describes how to install the various resources needed to run a CICS region, such as control tables and user application programs.

Part 2. Defining data sets ... pages 91—212
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 3. System initialization ... pages 213—360
Describes the system initialization parameters. that you can code to initialize a CICS region tailored for your installation.

Part 4. Initializing CICS data sharing servers... pages 363—413
Describes how to set up and start CICS data sharing servers.

Bibliography

CICS Transaction Server for OS/390

CICS Transaction Server for OS/390: Planning for Installation	GC33-1789
CICS Transaction Server for OS/390 Release Guide	GC34-5352
CICS Transaction Server for OS/390 Migration Guide	GC34-5353
CICS Transaction Server for OS/390 Installation Guide	GC33-1681
CICS Transaction Server for OS/390 Program Directory	GI10-2506
CICS Transaction Server for OS/390 Licensed Program Specification	GC33-1707

CICS books for CICS Transaction Server for OS/390

General	
CICS Master Index	SC33-1704
CICS User's Handbook	SX33-6104
CICS Transaction Server for OS/390 Glossary (softcopy only)	GC33-1705
Administration	
CICS System Definition Guide	SC33-1682
CICS Customization Guide	SC33-1683
CICS Resource Definition Guide	SC33-1684
CICS Operations and Utilities Guide	SC33-1685
CICS Supplied Transactions	SC33-1686
Programming	
CICS Application Programming Guide	SC33-1687
CICS Application Programming Reference	SC33-1688
CICS System Programming Reference	SC33-1689
CICS Front End Programming Interface User's Guide	SC33-1692
CICS C++ OO Class Libraries	SC34-5455
CICS Distributed Transaction Programming Guide	SC33-1691
CICS Business Transaction Services	SC34-5268
Diagnosis	
CICS Problem Determination Guide	GC33-1693
CICS Messages and Codes	GC33-1694
CICS Diagnosis Reference	LY33-6088
CICS Data Areas	LY33-6089
CICS Trace Entries	SC34-5446
CICS Supplementary Data Areas	LY33-6090
Communication	
CICS Intercommunication Guide	SC33-1695
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
CICS External Interfaces Guide	SC33-1944
CICS Internet Guide	SC34-5445
Special topics	
CICS Recovery and Restart Guide	SC33-1698
CICS Performance Guide	SC33-1699
CICS IMS Database Control Guide	SC33-1700
CICS RACF Security Guide	SC33-1701
CICS Shared Data Tables Guide	SC33-1702
CICS Transaction Affinities Utility Guide	SC33-1777
CICS DB2 Guide	SC33-1939

CICSPlex SM books for CICS Transaction Server for OS/390

General	
CICSPlex SM Master Index	SC33-1812
CICSPlex SM Concepts and Planning	GC33-0786
CICSPlex SM User Interface Guide	SC33-0788
CICSPlex SM View Commands Reference Summary	SX33-6099
Administration and Management	
CICSPlex SM Administration	SC34-5401
CICSPlex SM Operations Views Reference	SC33-0789
CICSPlex SM Monitor Views Reference	SC34-5402
CICSPlex SM Managing Workloads	SC33-1807
CICSPlex SM Managing Resource Usage	SC33-1808
CICSPlex SM Managing Business Applications	SC33-1809
Programming	
CICSPlex SM Application Programming Guide	SC34-5457
CICSPlex SM Application Programming Reference	SC34-5458
Diagnosis	
CICSPlex SM Resource Tables Reference	SC33-1220
CICSPlex SM Messages and Codes	GC33-0790
CICSPlex SM Problem Determination	GC33-0791

Other CICS books

CICS Application Programming Primer (VS COBOL II)	SC33-0674
CICS Application Migration Aid Guide	SC33-0768
CICS Family: API Structure	SC33-1007
CICS Family: Client/Server Programming	SC33-1435
CICS Family: General Information	GC33-0155
CICS 4.1 Sample Applications Guide	SC33-1173
CICS/ESA 3.3 XRF Guide	SC33-0661

If you have any questions about the CICS Transaction Server for OS/390 library, see *CICS Transaction Server for OS/390: Planning for Installation* which discusses both hardcopy and softcopy books and the ways that the books can be ordered.

Books from related libraries

Systems Network Architecture (SNA)

- *Systems Network Architecture - Function Description of Logical Unit Types*, GC20-1868.
- *Systems Network Architecture - Types of Logical Unit to Logical Unit Sessions*, GC20-1869.

Systems Application Architecture (SAA)

- *IBM System Application Architecture Common Programming Interface Reference Manual*, SC09-1308

Advanced communications function for VTAM (ACF/VTAM)

- *Network Program Products General Information*, GC30-3350
- *Advanced Communications Function for VTAM Installation and Resource Definition*, SC23-0111.
- *Advanced Communications Function for VTAM Customization*, SC23-0112.

- *Advanced Communications Function for VTAM Operation*, SC23-0113.
- *Advanced Communications Function for VTAM Messages and Codes*, SC23-0114.
- *Advanced Communications Function for VTAM Diagnosis Guide*, SC23-0116.
- *Advanced Communications Function for VTAM Diagnosis Reference*, LY30-5582.
- *Advanced Communications Function for VTAM Data Areas*, LY30-5584.
- *Advanced Communications Function for VTAM Programming*, SC23-0115.
- *Advanced Communications Function for VTAM Reference Summary*, SC23-0135.

Telecommunications Access Method (TCAM)

- *TCAM Concepts and Applications*, GC30-2049
- *TCAM System Programmer's Guide,. TCAM Level 10*, GC30-2051
- *TCAM Application Programmer's Guide,. TCAM Level 10*, GC30-3036.
- *TCAM Macro Reference Guide, TCAM Level 10*, GC30-2052

Virtual Storage Access Method (VSAM)

- *MVS/ESA Integrated Catalog Administration: Access Method Services Reference*, GC26-4135.
- *MVS/ESA Catalog Administration Guide*, GC26-4138
- *MVS/ESA Data Administration Guide*, GC26-4140
- *MVS/ESA VSAM Administration Guide*, GC26-4151
- *MVS/ESA VSAM Administration: Macro Instruction Reference*, GC26-4152.
- *MVS/ESA Catalog User's Guide*,. GC26-4041

DATABASE 2 (DB2)

- *Application Programming and SQL Guide*, SC26-4377
- *Administration Guide*, SC26-4374
- *Command and Utility Reference*, SC26-4378

Programming language support

- *VS COBOL II Application Programming Guide*, SC26-4045
- *VS COBOL II Installation and Customization* manual,. SC26-4048
- *OS PL/I Version 2 Installation and Customization under MVS Guide*, SC26-4311
- *IBM C/370 Installation Guide*, GC09-1331
- *IBM C/370 Programming Guide*, SC09-1384
- *Language Environment Programming Guide*, SC26-4818
- *Language Environment Planning for Installation and Customization Guide*, SC26-4817

Information Management System (IMS)

IMS/ESA Versions 4, 5, and 6 libraries

Table 1. IMS/ESA libraries

Title	Version 4	Version 5	Version 6
Administration Guide: Database Manager	----	SC26-8012	SC26-8725
Administration Guide: System	----	SC26-8013	SC26-8730

Table 1. IMS/ESA libraries (continued)

Title	Version 4	Version 5	Version 6
Administration Guide: Transaction Manager	----	SC26-8014	SC26-8731
Application Programming: Database Manager	----	SC26-8015	SC26-8727
Application Programming: Database Manager Summary	----	SC26-8037	----
Application Programming: DC Calls	SC26-4283	----	----
Application Programming: Design Guide	SC26-4279	SC26-8016	SC26-8728
Application Programming: DL/I Calls	SC26-4274	----	----
Application Programming: EXEC DLI Commands	SC26-4280	SC26-8018	SC26-8726
Application Programming: DL/I Calls Summary	SX26-3765	----	----
Application Programming: EXEC DLI Commands Summary	SX26-3775	SC26-8036	----
Application Programming: Transaction Manager	----	SC26-8017	SC26-8729
Application Programming: Transaction Manager Summary	----	SC26-8038	----
Customization Guide	----	SC26-8020	SC26-8732
Common Queue Server Reference	----	----	LY37-3730
Customization Guide: Database	SC26-4624	----	----
Customization Guide: Data Communications	SC26-4625	----	----
Customization Guide: Systems	SC26-4285	----	----
Data Communication Administration Guide	SC26-4286	----	----
Database Administration Guide	SC26-4281	----	----
Database Recovery Control Guide and Reference	----	----	SC26-8733
Diagnosis Guide and Reference	LY27-9539	LY27-9620	LY37-3731
Failure Analysis Structure Tables (FAST) for Dump Analysis	LY27-9512	LY27-9621	LY37-3732
General Information	GC26-4275	GC26-3467	----
Installation Guide	SC26-4276	----	----
Installation Volume 1: Installation and Verification	----	SC26-8023	GC26-8736
Installation Volume 2: System Definition and Tailoring	----	SC26-8024	GC26-8737
Licensed Programming Specifications	----	GC26-8040	GC26-8738
LU6.1 Adapter for LU6.2 Applications: Program Description/Operations	SC26-4392	----	----
Master Index and Glossary	SC26-4291	SC26-8027	----
Messages and Codes	SC26-4290	SC26-8028	GC26-8739
Open Transaction Manager Access Guide/Reference	----	SC26-8026	SC26-8743
Operations Guide	SC26-4287	SC26-8029	SC26-8741
Operator's Reference	SC26-4288	SC26-8030	SC26-8742
Release Planning Guide	GC26-4386	GC26-8031	GC26-8744

Table 1. IMS/ESA libraries (continued)

Title	Version 4	Version 5	Version 6
Sample Operating Procedures	SC26-4277	SC26-8032	SC26-8767
Summary of Operator Commands/Summary of. Commands	SX26-3764	SC26-8042	SC26-8766
System Administration Guide	SC26-4282	----	----
System Definition Reference	SC26-4278	----	----
Utilities Reference: Database/Utilities Reference:. Database Manager	SC26-4627	SC26-8034	SC26-8769
Utilities Reference: Data Communication	SC26-4628	----	----
Utilities Reference: Systems	SC26-4629	SC26-8035	SC26-8770
Utilities Reference: Transaction Manager	----	SC26-8022	SC26-8771

IBM CICS VSAM Recovery MVS/ESA

- *IBM CICS VSAM Recovery MVS/ESA General Information*, GH19-6707.
- *IBM CICS VSAM Recovery MVS/ESA User's Guide. and Reference*, SH19-6970.
- *IBM CICS VSAM Recovery MVS/ESA Implementation Guide*, SH19-6971.
- *IBM CICS VSAM Recovery MVS/ESA Messages and Problem Determination*, SH19-4006.

MVS

- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS Initialization and Tuning Guide*, SC28-1751
- *OS/390 MVS Installation Exits*, SC28-1753
- *OS/390 MVS Conversion Notebook*, GC28-1747
- *OS/390 MVS System Commands*, GC28-1781
- *OS/390 MVS System Management Facilities (SMF)*, GC28-1783
- *OS/390 MVS Diagnosis: Procedures*, SY28-1082
- *OS/390 MVS Diagnosis: Tools and Service Aids*, SY28-1085

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.

Summary of changes

This edition of the *CICS System Definition Guide* is based on the CICS Transaction Server for OS/390 Release 2 edition. Changes from CICS Transaction Server for OS/390 Release 2 are marked by a vertical line to the left of the change.

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

The main changes made to this book for CICS Transaction Server for OS/390 Release 3 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
- New SIT parameters
  - AICONS
  - DOCCODEPAGE
  - DSRTPGM
  - ENCRYPTION
  - FORCEQR
  - KEYFILE
  - MAXOPENTCBS
  - NCPLDFT
  - RRMS
  - RUWAPOOL
  - SSLDELAY
  - TCPIP

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

The main changes made to this book for the CICS Transaction Server for OS/390 Release 2 include:

- 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 12. Defining CICS log streams” on page 121.
- The SYSLOG system initialization parameter, used to preserve data on the system log in CICS Transaction Server for OS/390 Release 2,. is obsolete in CICS Transaction Server for OS/390 Release 3.
- 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 SIT parameters:
  - DB2CONN
  - DBCTLCON

  - WEB
  - WEBDELAY

Changes for the CICS Transaction Server for OS/390 release 1 edition

The main changes made to this book for CICS Transaction Server for OS/390 release 1 include:

- The following new SIT parameters:
  - SYDUMAX
  - TRDUMAX
  - CSDINTEG
  - CSDRLS
  - OFFSITE
  - RLS
  - SDTRAN
  - SYSLOG
  - TDINTRA
- Changes to the following SIT parameters:
  - AILDELAY
  - AIRDELAY
  - DCT
  - START=INITIAL
- Resource definition online for transient data destinations.
- Resource definition online for journalmodels.
- An additional section in “Chapter 10. Defining the temporary storage data set” on page 107, 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 26. Starting up temporary storage servers” on page 367.

Part 1. Installing resource definitions

After you have installed CICS, as described in the *CICS Transaction Server for OS/390 Installation Guide*, you need to install all the resources that your CICS regions need to run user transactions, and define those resources to CICS. This section of the book describes how to install resources and resource definitions in a CICS region.

It consists of the following chapters:

Chapter 1. Introducing resource definition

Before you can use CICS, you must supply it with information about the resources it should use, and how it should 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 are to interact with each other.

You supply this information to CICS by using one or more of the following methods of **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 are 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 means of a batch job submitted 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 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* manual.
5. **Macro definition**: You can use assembler macro source to define resources. Definitions are stored in assembled tables in a program library, from which they are installed during CICS initialization.

Which methods you use depends on the resources you want to define. Table 2 on page 4 suggests some of the things you should consider when deciding which definition method to use when there is a choice.

Table 2. 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.	• It enables configuration and installation of CICS resources for large numbers of CICS regions from a single management focal point. • It also allows you to write applications for administering the running CICS system.	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.	• You can modify or define a large number of resources in one job. • You can run DFHCSDUP against a non-recoverable CSD file while it is being shared between CICS regions using RLS access mode.	• You cannot install resources into an active CICS system. • You cannot make updates via DFHCSDUP against a recoverable CSD file that is being accessed in RLS mode.
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	• Using this method, you code and assemble macroinstructions to define resources in the form of tables. • This method is available only for a limited number of resource types.	None. However, this method must be used when no other methods are available.	• 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. • You must do time-consuming assemblies to generate macro tables.

For information about CICS resource definition, 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 required in a particular run of CICS by a **list** of groups. You can specify up to four lists of groups to be installed during CICS initialization, using the GRPLIST=listname system initialization parameter. You can also use the CEDA INSTALL command to install a resource definition or group of definitions 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 modify 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, you can:

- Protect groups of resources by using the CEDA command LOCK
- Protect the list of resource groups 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 that cannot be defined in the CSD. The tables and their resource definitions are created by using the CICS table assembly macro instructions. 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 “Chapter 2. Defining resources in CICS control tables” on page 7.

Using the CSD and control tables together

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

1. On an initial or cold start, you can mix file control resources 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. Definitions other than for BDAM will not be installed. Such definitions are still supported because the DFHFCT macro is used by the utility that migrates definitions to the CSD. 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 [1] terminals defined in a TCT with resource definitions for VTAM terminals 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.
3. Although you can use a mixture of RDO and macro definitions for transient data resources, you are recommended to use RDO. This is because there are features in RDO that are not available if you use the DFHDCT macro. For example, the INDOUBT attributes, WAIT and WAITACTION. Furthermore, support for the DFHDCT macro will be withdrawn in a future release of CICS.

   When CICS is fully initialized, additional transient data entries can be added using CEDA. On a warm or an emergency restart, the DCT is reconstructed

1. Non-VTAM terminals. TCT entries can be for TCAM DCB terminals, BSAM sequential devices, logical device codes (LDCs), and remote BTAM terminals required for ISC/MRO purposes.

from the global catalog. On an initial or cold start, avoid duplicate transient data queue names. Duplicate names result in the production of error messages, and CICS continues to use the definition that was installed first.

Chapter 2. Defining resources in CICS control tables

This chapter describes what you should do to define resource definitions in CICS control tables. The tables and their resource definitions are created by using macros. You must use macros to define non-VTAM networks and terminals, non-VSAM files, databases, and resources for monitoring and system recovery. Queues can be defined using both RDO and macro definition. For details about defining resource definitions in CICS control tables, and about migrating your tables to the CSD, see, TYPETERM, TERMINAL and PROFILE definitions, in the *CICS Resource Definition Guide* .

For each of the CICS tables listed on page Table 3 on page 8, complete the following steps:

1. Code the resource definitions you require.
2. Assemble and link-edit these definitions, using the CICS-supplied procedure DFHAUPLE, to create a load module in the required CICS load library. The load library is either CICSTS13.CICS.SDFHLOAD or CICSTS13.CICS.SDFHAUTH, which you must specify by the NAME parameter of the DFHAUPLE procedure. The CICS-supplied macros used to create the CICS tables determine whether tables are loaded above the 16MB line. All tables, other than the TCT, are loaded above the 16MB line.
3. Name the suffix of the load module by a system initialization parameter. For most of the CICS tables, if you do not require the table you can code *tablename=NO*. The exceptions to this rule are as follows:
   - The CLT. Specifying CLT=NO causes CICS to try and load DFHCLTNO. The CLT is used only in the alternate CICS, when you are running CICS with XRF, and is always required in that case.
   - The SIT. Specifying SIT=NO causes CICS to try and load DFHSITNO. The SIT is always needed, and you can specify the suffix by coding the SIT system initialization parameter.
   - The TCT. Specifying TCT=NO causes CICS to load a dummy TCT named DFHTCTDY, as explained on page 318.
   - The TLT. Terminal list tables are specified by program entries in the CSD, and do not have a system initialization parameter.
   - The MCT. Specifying MCT=NO causes the CICS monitoring domain to build 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 (or classes) are active.
4. If you are running CICS with XRF, the active and the alternate CICS regions share the same versions of tables. However, to provide protection against DASD failure, you might want to run your active and alternate CICS regions from separate sets of load libraries—in which case, you should make the separate copies **after** generating your control tables.

Table 3 lists all the CICS tables that can be assembled, link-edited, and installed in your CICS libraries for use in your CICS system.

Table 3. CICS tables that you can assemble, link-edit, and install in CICS libraries

Table	Module Name	Abbreviation	Required load library
Command list table	DFHCLTxx	CLT	SDFHAUTH
Data conversion table	DFHCNV	CNV	SDFHLOAD
Destination control table	DFHDCTxx	DCT	SDFHLOAD
File control table	DFHFCTxx	FCT	SDFHLOAD
Monitor control table	DFHMCTxx	MCT	SDFHLOAD
Program list table	DFHPLTxx	PLT	SDFHLOAD
DL/I program specification block	DFHPSBxx	PSB	SDFHLOAD
Recoverable service element table	DFHRSTxx	RST	SDFHAUTH
System initialization table	DFHSITxx	SIT	SDFHAUTH
System recovery table	DFHSRTxx	SRT	SDFHLOAD
Terminal control table	DFHTCTxx	TCT	SDFHLOAD
Terminal list table	DFHTLTxx	TLT	SDFHLOAD
Temporary storage table	DFHTSTxx	TST	SDFHLOAD
Transaction list table	DFHXLTxx	XLT	SDFHLOAD

You can generate several versions of each CICS control table by specifying SUFFIX=xx in the macro that generates the table. This suffix is then appended to the default 6-character name of the load module.

To get you started, CICS provides the sample tables listed in Table 4 in the CICSTS13.CICS.SDFHSAMP library:

Table 4. Sample CICS system tables in the CICSTS13.CICS.SDFHSAMP library

Table	Suffix	Notes
Command list table (CLT)	1$	XRF regions only
Monitor control table (MCT)	A$	For a CICS AOR
Monitor control table (MCT)	F$	For a CICS FOR
Monitor control table (MCT)	T$	For a CICS TOR
Monitor control table (MCT)	2$	
System initialization table (SIT)	$$	Default system initialization parameters
System initialization table (SIT)	6$	
System recovery table (SRT)	1$	
Terminal control table (TCT)	5$	Non-VTAM terminals only

Unless you have TCAM terminals or are using sequential devices, you do not need a TCT and should specify TCT=NO. (For information about the effect of TCT=NO, see page 318.) You define VTAM terminals in the CSD only, either explicitly or by means of autoinstall model definitions but, for non-VTAM terminals, you must use DFHTCT macros.

Although you can modify and reassemble the tables while CICS is running, you must shut down and restart CICS to make the new tables available to CICS. (The command list table (CLT) is an exception in that a new table can be brought into use without shutting down either the active CICS region or the alternate CICS region.)

Defining control tables to CICS

You can assemble and link-edit more than one version of a table, and (except for the CNV) use a suffix to distinguish them. To specify which version you want CICS to use, you code a system initialization parameter of the form *tablename=xx*. (For example, TCT=5$.) However, you can code the SIT=xx parameter only as a startup override; that is, not in the DFHSIT table. For details of all the CICS system initialization parameters, see “Chapter 21. CICS system initialization parameters” on page 215.

Other tables that have special requirements are program list tables (PLTs), terminal list tables (TLTs), and transaction list tables (XLTs). For each TLT, autoinstall for programs must be active or you must specify a program resource definition in the CSD, using the table name (including the suffix) as the program name. PLTs or XLTs are autoinstalled if there is no program resource definition in the CSD. For example, to generate a TLT with a suffix of AA (DFHTLTAA), the CEDA command would be as follows:

```
CEDA DEFINE PROGRAM(DFHTLTAA) GROUP(grpname) LANGUAGE(ASSEMBLER)
```

For information about program and terminal list tables, see the *CICS Resource Definition Guide* .

The DFHCNV conversion table is required when communicating with CICS on a non-System 390 platform. For information about the Data Conversion Process, see the *CICS Family: Communicating from CICS on System/390*.

The command list table (CLT) is used only by the alternate CICS in a CICS system running with XRF=YES, and differs in many other respects from the other CICS tables. For more guidance information, including information on resource definition specific to the CICS extended recovery facility, see the *CICS/ESA 3.3 XRF Guide*.

Assembling and link-editing control tables: the DFHAUPLE procedure

To assemble and link-edit your tables, write a job that invokes the CICS-supplied sample procedure DFHAUPLE. The DFHAUPLE procedure needs the following libraries to be online:

Table 5. Library requirements for the DFHAUPLE procedure

Library name	Tables required for
SYS1.MACLIB	All
CICSTS13.CICS.SDFHMAC	All
CICSTS13.CICS.SDFHLOAD	All except the CLT, RST, and SIT
CICSTS13.CICS.SDFHAUTH	CLT, RST, and SIT
SMP/E global zone	All
SMP/E target zone	All

When you have assembled and link-edited your tables, define them to CICS by system initialization parameters.

Steps in the DFHAUPLE procedure

The DFHAUPLE procedure, shown in Figure 1 on page 11, is tailored to your CICS environment and stored in the CICSTS12.XDFHINST library when you run the DFHISTAR job. (For information about DFHISTAR, see the *CICS Transaction Server for OS/390 Installation Guide*.) It consists of the following steps:

1. ASSEM: This step puts your table definition macros into a temporary partitioned data set (PDS) member that is used as input to the ASM and SMP steps. The BLDMBR step subsequently adds further members to this temporary PDS.
2. ASM: In this assembly step, SYSPUNCH output is directed to a temporary sequential data set. This output consists of IEBUPDTE control statements, linkage editor control statements, SMP control statements, and the object deck.
3. BLDMBR: In this step, the IEBUPDTE utility adds further members to the temporary PDS created in the ASSEM step. These members contain linkage editor control statements and SMP control statements, and the object deck from the assembly step.
4. LNKEDT: The link-edit step uses the contents of the PDS member LNKCTL as control statements. The object code produced in step 2 is included from the temporary PDS. The output is placed in the load library specified by the NAME parameter on the procedure. You must specify NAME=SDFHAUTH for the CLT, RST, and SIT, and NAME=SDFHLOAD for all the others.
5. ZNAME: This step creates a temporary data set, which is passed to the SMP/E JCLIN job step, containing a SET BDY command defining a target zone name. This tells SMP/E which target zone to update.
6. SMP: The SMP step uses the temporary PDS members MACROS, SMPCNTL, SMPJCL1, SMPJCL2, LNKCTL, SMPEOF, and the object deck to update the control data set (CDS).
7. DELTEMP: This final step deletes the temporary partitioned data set, &&TEMPPDS.

   This step must run successfully if you want SMP to reassemble CICS tables automatically when applying later maintenance.

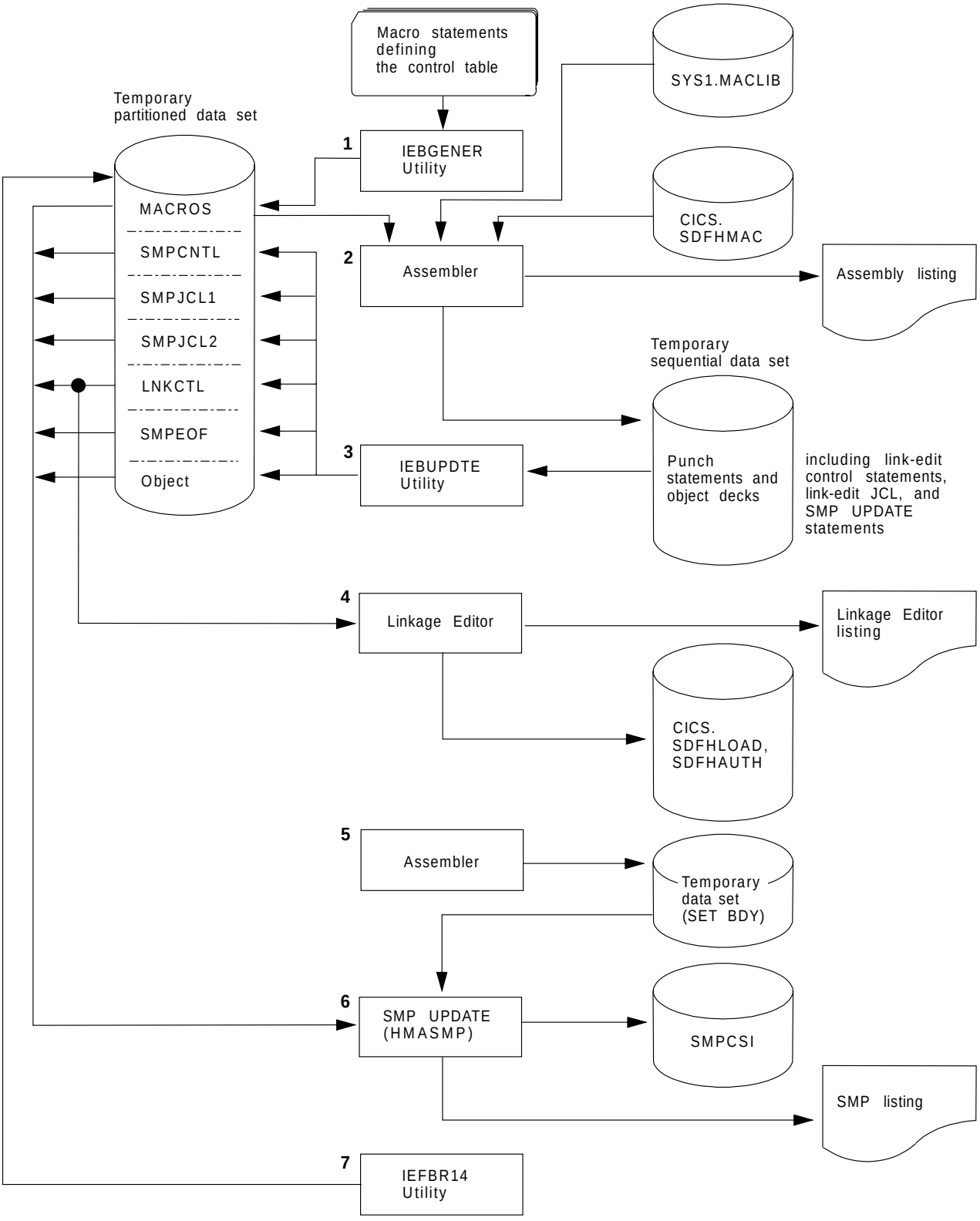


Figure 1. Assembling and link-editing the control tables using the DFHAUPLE procedure

Sample job stream for non-XRF tables

You can use the following job stream to assemble and link-edit all of the CICS control tables except for the CLT and the RST:

```
//jobname      JOB     (accounting information),
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//ASMTAB       EXEC    PROC=DFHAUPLE[,NAME={SDFHLOAD|SDFHAUTH}]
//*
//ASSEM.SYSUT1 DD   *
        .
      Control table macro statements
        .
/*
```

Figure 2. Job stream for non-XRF tables

Note: Specify NAME=SDFHAUTH on this job for the system initialization table only; link-edit all other tables (except the CLT and RST) into SDFHLOAD.

Sample job stream for XRF-related tables

If you use CICS with XRF, there are two other tables that are assembled and link-edited using the DFHAUPLE procedure. These are:

CLT The command list table. This gives a list of MVS system commands and messages to the operator that CICS issues if a takeover occurs.

RST The recoverable service table. 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.

Both of these tables must be link-edited, with the reentrant attribute, into an APF authorized library. You specify that the table is reentrant on the RENTATT parameter of the DFHAUPLE procedure. A sample job to call the DFHAUPLE procedure for the CLT and RST is as follows:

```
//jobname      JOB     (accounting information),
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//             EXEC    PROC=DFHAUPLE,NAME=SDFHAUTH,RENTATT=RENT
//*
//ASSEM.SYSUT1 DD   *
        .
      Control table macro statements
        .
/*
```

Figure 3. Job stream for XRF-related tables

Chapter 3. Installing map sets and partition sets

This chapter describes how to assemble and link-edit map sets and partition sets for use with the basic mapping (BMS) facility of CICS, assuming that they are defined by CICS-supplied macros.

CICS also supports the definition of BMS map sets and partition sets interactively by using licensed programs such as the IBM Screen Definition Facility II (SDF II), program number 5665-366.

For information about writing programs to use BMS services, see the *CICS Application Programming Guide*.

CICS loads BMS map sets and partition sets above the 16MB line if you specify the residency mode for the map set or partition set as RMODE(ANY) in the link-edit step. If you are using either map sets or partition sets from earlier releases of CICS, you can load them above the 16MB line by link-editing them again with RMODE(ANY). For examples of link-edit steps specifying RMODE(ANY), see the sample job streams in this chapter.

CICS provides the DFHASMVS procedure (installed in the CICSTS13.CICS.SDFHPROC library) that you can use to assemble and link-edit map sets.

Note: The DFHASMVS procedure refers to the MVS library SYS1.MODGEN. If you have not yet restructured MVS (moving members from SYS1.AMODGEN to SYS1.MODGEN), change the SYS1.MODGEN reference to SYS1.AMODGEN in the DFHASMVS procedure, until you have restructured MVS. When you have restructured MVS, you must return the SYS1.AMODGEN reference to SYS1.MODGEN.

Installing map sets

This section first describes the types of map sets, how you define them, and how CICS recognizes them. This is followed by a description of how to prepare physical map sets and symbolic description map sets separately. Finally, there is a description of how to prepare both physical and symbolic description map sets in one job. In these descriptions, it is assumed that the SYSPARM parameter is used to distinguish the two types of map sets.

Types of map sets

To install a map set, you must actually prepare two types of map sets:

- A **physical** map set, used by BMS to translate data from the standard device-independent form used by application programs to the device-dependent form required by terminals.
- A **symbolic description** map set, used in the application program to define the standard device-independent form of the user data. This is a DSECT in assembler language, a data definition in COBOL, a BASED or AUTOMATIC structure in PL/I, and a “struct” in C/370.

Physical map sets must be cataloged in the CICS load library. Symbolic description map sets can be cataloged in a user copy library, or inserted directly into the application program itself.

The map set definition macros are assembled twice; once to produce the physical map set used by BMS in its formatting activities, and once to produce the symbolic description map set that is copied into the application program.

Defining the type of map set you require

The two types of map set can be distinguished by either:

- The TYPE operand of the DFHMSD macro
- Use of the SYSPARM operand on the EXEC statement of the job used to assemble the map set

If you use the SYSPARM operand for this purpose, the TYPE operand of the DFHMSD macro is ignored. Using SYSPARM allows both the physical map set and the symbolic description map set to be generated from the same unchanged set of BMS map set definition macros.

Map sets can be assembled as either **unaligned** or **aligned** (an aligned map is one in which the length field is aligned on a halfword boundary). Use unaligned maps except in cases where an application package needs to use aligned maps.

The SYSPARM value alone determines whether the map set is aligned or unaligned, and is specified on the EXEC PROC=DFHASMVS statement. The TYPE operand of the DFHMSD macro can only define whether a physical or symbolic description map set is required. The SYSPARM operand can also be used to specify whether a physical map set or a symbolic description map set (DSECT) is to be assembled, in which case it overrides the TYPE operand. If neither operand is specified, an unaligned DSECT is generated.

For the possible combinations of operands to generate the various types of map set, see Table 6.

Table 6. SYSPARM and DFHMSD operand combinations for map assembly

Type of map set	SYSPARM operand of EXEC DFHASMVS statement	TYPE operand of DFHMSD macro
Aligned symbolic description map set (DSECT)	A A ADSECT	Not specified DSECT Any (takes SYSPARM)
Aligned physical map set	A AMAP	MAP Any (takes SYSPARM)
Unaligned symbolic description map set (DSECT)	Not specified Not specified DSECT	Not specified DSECT Any (takes SYSPARM)
Unaligned physical map set	Not specified MAP	MAP Any (takes SYSPARM)

The physical map set indicates whether it was assembled for aligned or unaligned maps. This information is tested at execution time, and the appropriate map alignment used. Thus aligned and unaligned map sets can be mixed.

Using extended data stream terminals

Applications and maps designed for the 3270 Information Display System run unchanged on devices supporting extensions to the 3270 data stream such as color, extended highlighting, programmed symbols, and validation. To use fixed extended attributes such as color, you only need to reassemble the physical map set. If dynamic attribute modification by the application program is needed, you must reassemble both the physical and symbolic description map sets, and you must reassemble or recompile the application program.

Installing physical map sets

Figure 4 shows the procedure for installing physical map sets.

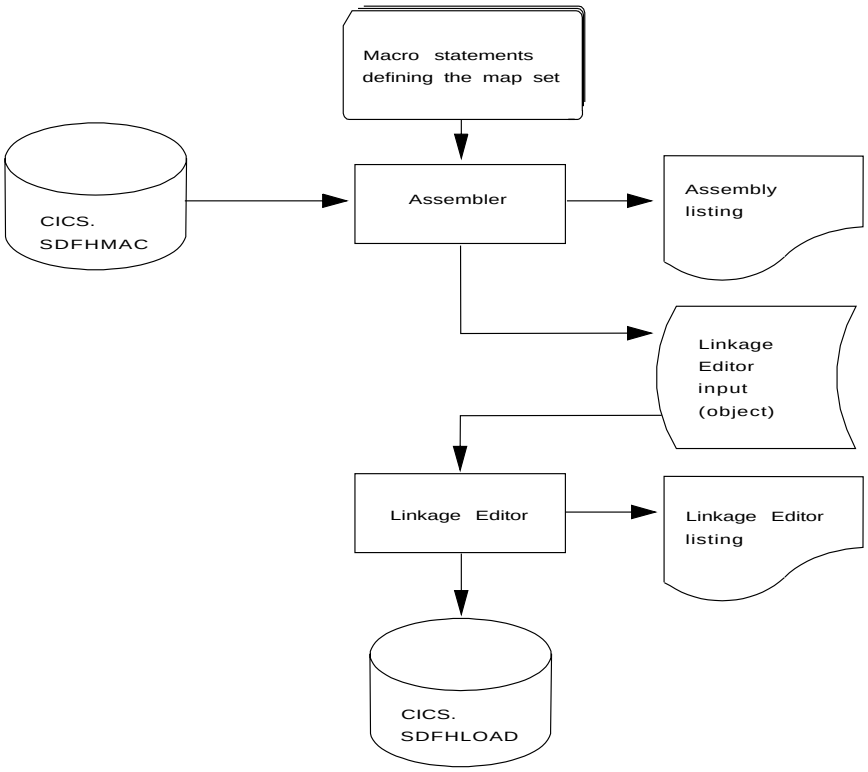


Figure 4. Installing physical map sets

Figure 5 on page 16 gives an example job stream for the assembly and link-editing of physical map sets.

```
//PREP     JOB  'accounting information',CLASS=A,MSGLEVEL=1
//STEP1    EXEC PROC=DFHASMVS,PARM.ASSEM='SYSPARM(MAP)'          1
//SYSPUNCH DD   DSN=&&TEMP,DCB=(RECFM=FB,BLKSIZE=2960),
//         SPACE=(2960,(10,10)),UNIT=SYSDA,DISP=(NEW,PASS)
//SYSIN    DD   *
   ⋮
      Macro statements defining the map set
   ⋮
/*
//STEP2    EXEC PROC=DFHLNKVS,PARM='LIST,LET,XREF'               2
//SYSLIN   DD   DSN=&&TEMP,DISP=(OLD,DELETE)
//         DD   *
           MODE RMODE(ANY|24)                                    3
           NAME mapsetname(R)                                    4
/*
//
```

Figure 5. Assembling and link-editing a physical map set

Notes:

1 For halfword-aligned length fields, specify the option SYSPARM(AMAP) instead of SYSPARM(MAP).

2 Physical map sets are loaded into CICS-key storage, unless they are link-edited with the RMODE(ANY) and RENT options. If they are link-edited with these options, they are loaded into key-0 protected storage, provided that RENTPGM=PROTECT is specified on the RENTPGM initialization parameter.

However, it is recommended that map sets should not be link-edited with the RENT or the REFR options because, in some cases, CICS modifies the map set.

For more information about the storage protection facilities available in CICS, see “Storage protection” on page 353.

3 The MODE statement specifies whether the map set is to be loaded above (RMODE(ANY)) or below (RMODE(24)) the 16MB line. RMODE(ANY) indicates that CICS can load the map set anywhere in virtual storage, but tries to load it above the 16MB line, if possible.

4 Use the NAME statement to specify the name of the physical map set that BMS loads into storage. If the map set is device-dependent, derive the map set name by appending the device suffix to the original 1- to 7-character map set name used in the application program. The suffixes to be appended for the various terminals supported by CICS BMS depend on the parameter specified in the TERM or SUFFIX operand of the DFHMSD macros used to define the map set. For programming information giving a complete list of map set suffixes, see the *CICS Application Programming Reference* manual.

To use a physical map set, you must define and install a resource definition for it. You can do this either by using the program autoinstall function or by using the CEDA DEFINE MAPSET and INSTALL commands, as described in “Defining programs, map sets, and partition sets to CICS” on page 56.

Installing symbolic description map sets

Symbolic description map sets enable the application programmer to make symbolic references to fields in the physical map set. Figure 6 shows the preparation of symbolic description map sets for BMS.

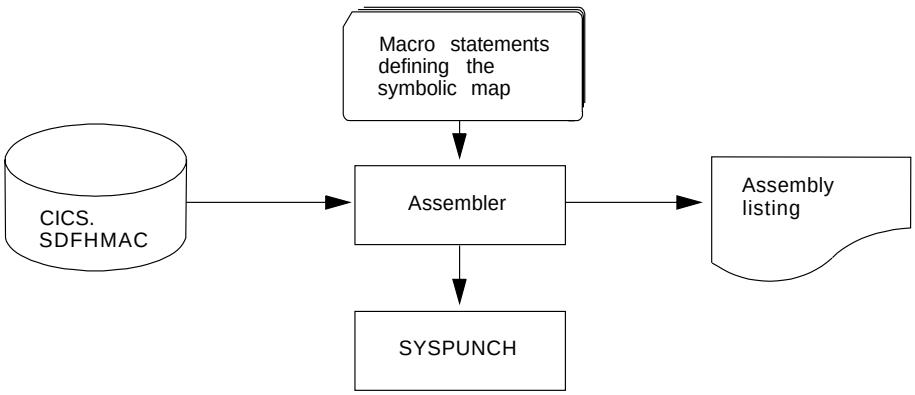


Figure 6. Installing symbolic description map sets using the DFHASMVS procedure

To use a symbolic description map set in a program, you must assemble the source statements for the map set and obtain a punched copy of the storage definition through SYSPUNCH. The first time this is done, you can direct the SYSPUNCH output to SYSOUT=A to get a listing of the symbolic description map set. If many map sets are to be used at your installation, or there are multiple users of common map sets, establish a private user copy library for each language that you use.

When a symbolic description is prepared under the same name for more than one programming language, a separate copy of the symbolic description map set must be placed in each user copy library. You must ensure that the user copy libraries are correctly concatenated with SYSLIB.

You need only one symbolic description map set corresponding to all the different suffixed versions of the physical map set. For example, to run the same application on terminals with different screen sizes, you would:

1. Define two map sets each with the same fields, but positioned to suit the screen sizes. Each map set has the same name but a different suffix, which would match the suffix specified for the terminal.
2. Assemble and link-edit the different physical map sets separately, but create only one symbolic description map set, because the symbolic description map set would be the same for all physical map sets.

You can use the sample job stream in Figure 7 to obtain a listing of a symbolic description map set. It applies to all the programming languages supported by CICS.

```
//DSECT    JOB  'accounting information',CLASS=A,MSGLEVEL=1
//ASM      EXEC PROC=DFHASMVS,PARM.ASSEM='SYSPARM(DSECT)'
//SYSPUNCH DD   SYSOUT=A
//SYSIN    DD   *
   ⋮
      Macro statements defining the map set
   ⋮
/*
//
```

Figure 7. Listing of a symbolic description map set

If you want to assemble symbolic description map sets in which length fields are halfword-aligned, change the EXEC statement of the sample job in Figure 7 to the following:

```
//ASSEM  EXEC PROC=DFHASMVS,PARM.ASSEM='SYSPARM(ADSECT)'
```

To obtain a punched copy of a symbolic description map set, code the //SYSPUNCH statement in the above example to direct output to the punch data stream. For example:

```
//SYSPUNCH DD   SYSOUT=B
```

To store a symbolic description map set in a private copy library, use job control statements similar to the following:

```
//SYSPUNCH DD   DSN=USER.MAPLIB.ASM(map set name),DISP=OLD
//SYSPUNCH DD   DSN=USER.MAPLIB.COB(map set name),DISP=OLD
//SYSPUNCH DD   DSN=USER.MAPLIB.PLI(map set name),DISP=OLD
```

Installing physical and symbolic description maps together

Figure 8 on page 19 shows the DFHMAPS procedure for installing physical and symbolic description maps together. The DFHMAPS procedure consists of the following four steps, shown in Figure 8:

1. The BMS macros that you coded for the map set are added to a temporary sequential data set.
2. The macros are assembled to create the physical map set. The MAP option is coded in the SYSPARM global variable in the EXEC statement (PARM='SYSPARM(MAP)').
3. The physical map set is link-edited to the CICS load library.
4. Finally, the macros are assembled again, this time to produce the symbolic description map set. In this step, DSECT is coded in the SYSPARM global variable in the EXEC statement (PARM='SYSPARM(DSECT)'). Output is directed to the destination specified in the //SYSPUNCH DD statement. In the DFHMAPS procedure, that destination is the CICSTS13.CICS.SDFHMAC library.

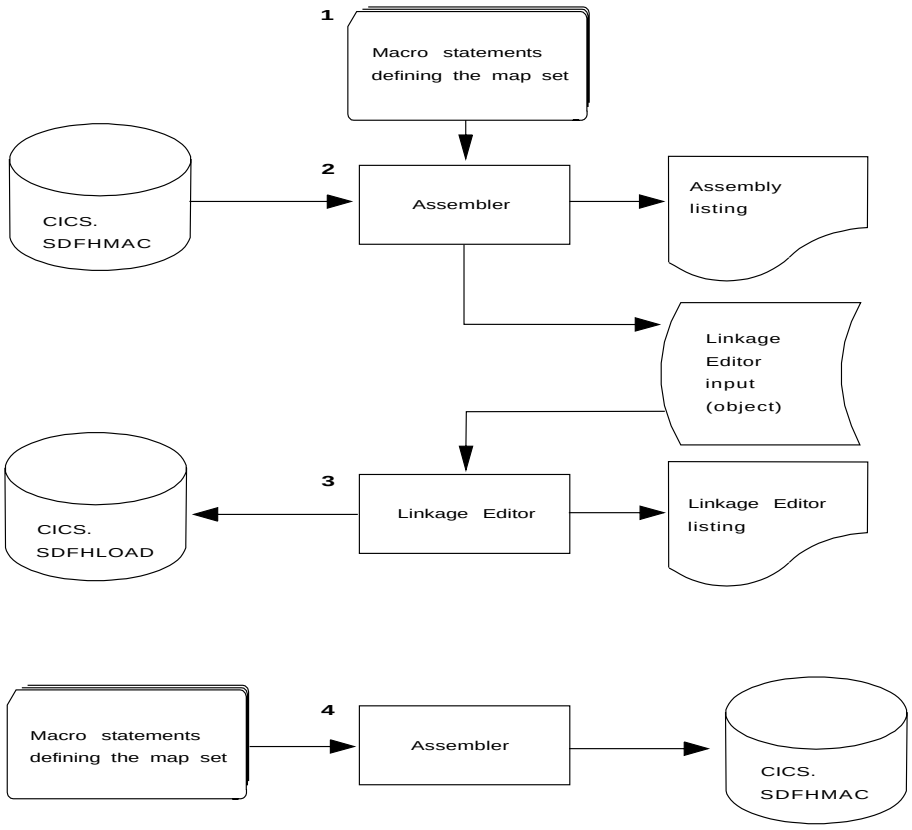


Figure 8. Installing a physical map set and a symbolic description map set together

JCL to install physical and symbolic description maps

The load module from the assembly of the physical map set and the source statements for the symbolic description map set can be produced in the same job by using the sample job stream in Figure 9.

```
//PREPARE  JOB  'accounting information',CLASS=A,MSGLEVEL=1
//ASSEM    EXEC PROC=DFHMAPS,MAPNAME=mapsetname,RMODE=ANY|24   (see note)
//SYSUT1   DD   *
   ⋮
      Macro statements defining the map set
   ⋮
/*
//
```

Figure 9. Installing physical and symbolic description maps together

Note: The RMODE statement specifies whether the map set is to be loaded above (RMODE=ANY) or below (RMODE=24) the 16MB line. RMODE=ANY indicates that CICS can load the map set anywhere in virtual storage, but tries to load it above the 16MB line, if possible.

The DFHMAPS procedure produces map sets that are not halfword-aligned. If you want the length fields in input maps to be halfword-aligned, you have to code A=A on the EXEC statement. In the sample job in Figure 9, change the EXEC statement to:

```
//ASSEM  EXEC  PROC=DFHMAPS,MAPNAME=mapsetname,A=A
```

This change results in the SYSPARM operands in the assembly steps being altered to SYSPARM(AMAP) and SYSPARM(ADSECT) respectively.

The DFHMAPS procedure directs the symbolic description map set output (SYSPUNCH) to the CICSTS13.CICS.SDFHMAC library. Override this by specifying DSCTLIB=name on the EXEC statement, where “name” is the name of the chosen user copy library.

Installing partition sets

Partition sets are installed in the same way as physical map sets (as illustrated in Figure 4 on page 15). There is no concept of a symbolic description partition set.

The job stream in Figure 10 is an example of the assembly and link-edit of partition sets.

```
//PREP     JOB  'accounting information',CLASS=A,MSGLEVEL=1
//STEP1    EXEC PROC=DFHASMVS
//SYSPUNCH DD   DSN=&&TEMP,DCB=(RECFM=FB,BLKSIZE=2960),
//         SPACE=(2960,(10,10)),UNIT=SYSDA,DISP=(NEW,PASS)
//SYSIN    DD   *
   ⋮
      Macro statements defining the partition set
   ⋮
/*
//STEP2    EXEC PROC=DFHLNKVS,PARM='LIST,LET,XREF'                 1
//SYSLIN   DD   DSN=&&TEMP,DISP=(OLD,DELETE)
//         DD   *
           MODE RMODE(ANY|24)                                      2
           NAME partitionsetname(R)                                3
/*
//
```

Figure 10. Assembling and link-editing a partition set

Notes:

1 A partition set is loaded into CICS-key storage, unless it is link-edited with the RMODE(ANY) and RENT options. If it is link-edited with these options, it is loaded into key-0 protected storage, provided that RENTPGM=PROTECT is specified on the RENTPGM initialization parameter.

For more information about the storage protection facilities available in CICS, see “Storage protection” on page 353.

2 The MODE statement specifies whether the partition set is to be loaded above (RMODE(ANY)) or below (RMODE(24)) the 16MB line. RMODE(ANY) indicates that CICS can load the partition set anywhere in virtual storage, but tries to load it above the 16MB line, if possible.

3 Use the NAME statement to specify the name of the partition set which BMS loads into storage. If the partition set is device-dependent, derive the partition set name by appending the device suffix to the original 1- to 7-character partition set name used in the application program. The suffixes that BMS appends for the various terminals depend on the parameter specified in the SUFFIX operand of the DFHPSD macro that defined the partition set.

For programming information giving a complete list of partition-set suffixes, see the *CICS Application Programming Reference* manual.

To use a partition set, you must define and install a resource definition for it. You can do this either by using the program autoinstall function or by using the CEDA DEFINE PARTITIONSET and INSTALL commands, as described in the *CICS Resource Definition Guide*.

Chapter 4. Installing application programs

This chapter describes what you have to do to install an application program to run under CICS. It provides the following information:

- "CICS-supplied facilities for installing programs" on page 24 describes the CICS-supplied procedures, translators, and interface modules that you can use to install application programs.
- "Adding support for programming languages" on page 26 describes what to do to add support for the programming languages that can be used with CICS. You should normally complete the actions in this section before installing your application programs.
- "Preparing to install application programs" on page 34 describes points that you should consider when preparing to install application programs and outlines the steps to install an application program to run under CICS.
- "Using the CICS-supplied procedures to install application programs" on page 41 describes how to use the CICS-supplied procedures to install application programs.
- "Using your own job streams" on page 53 describes important points that you should consider if you intend using your own JCL to install application programs.

In this chapter, **application program** generally means any user program that uses the CICS command-level application programming interface (API). Such programs can also use:

- SQL statements
- DLI requests
- Common programming interface (CPI) statements
- SAA Resource Recovery statements
- External CICS interface commands

For information about writing CICS application programs, see the *CICS Application Programming Guide*.

Note: If you are developing application programs to use the CICS dynamic transaction routing facility, you are recommended to use the Transaction Affinities Utility to detect whether the programs are likely to cause intertransaction affinity. See the *CICS Transaction Affinities Utility Guide* for more information about using the utility. See the *CICS Application Programming Guide* for a description of intertransaction affinity.

CICS-supplied facilities for installing programs

This section describes the CICS-supplied procedures, translators, and interface modules that you can use to install application programs.

The CICS-supplied procedures

CICS supplies job control statements (JCL) for the translate, compile, and link-edit steps, in separate cataloged procedures for each programming language supported. These procedures, installed in the CICSTS13.CICS.SDFHPROC library, should have been copied into a procedure library following installation of CICS. There is a list and brief description of these CICS-supplied procedures in the *CICS Transaction Server for OS/390 Program Directory*.

Each procedure has a name of the form DFHwxTyL, where the variables w, x, and y depend on the type of program (EXCI batch or CICS online), the type of compiler, and the programming language. Using the preceding naming convention, the procedure names are given in Table 7.

Table 7. Procedures for installing application programs

Language	**LE-conforming compilers**		**non-LE-conforming compilers**	
	Online	**EXCI**	**Online**	**EXCI**
Assembler	-	-	DFHEITAL	DFHEXTAL
C	DFHYITDL	DFHYXTDL	DFHEITDL	DFHEXTDL
C++	DFHYITEL	DFHYXTEL	-	-
COBOL	DFHYITVL	DFHYXTVL	DFHEITVL	DFHEXTVL
PL/I	DFHYITPL	DFHYXTPL	DFHEITPL	DFHEXTPL

The CICS-supplied translators

The following CICS-supplied translators are installed in the CICSTS13.CICS.SDFHLOAD library:

Assembler DFHEAP1$
C DFHEDP1$
COBOL DFHECP1$
PL/I DFHEPP1$

For a description of the translation process, and information about the translator options that you can specify, see Specifying translator options in the *CICS Application Programming Guide* .

Dynamic invocation of the translator

You can invoke the command-level language translator dynamically from a batch assembler-language program using an ATTACH, CALL, LINK, or XCTL macro; or from a C, PL/I, or COBOL program using CALL. If you use ATTACH, LINK, or XCTL, use the appropriate translator load module, DFHExP1$, where x=A for assembler language, x=C for COBOL, x=D for C, or x=P for PL/I.

If you use CALL, specify PREPROC as the entry point name to invoke the translator.

In all cases, pass the following address parameters to the translator:

- The address of the translator option list
- The address of a list of DD names used by the translator (this is optional)

These addresses must be in adjacent fullwords, aligned on a fullword boundary. Register 1 must point to the first address in the list, and the high-order bit of the last address must be set to one, to indicate the end of the list. This is the case whether one or two addresses are passed.

Translator option list: The translator option list must begin on a halfword boundary. The first two bytes contain a binary count of the number of bytes in the list (excluding the count field). The remainder of the list can contain any of the translator option keywords, separated by commas, blanks, or both.

Data definition (DD name) list: The DD name list must begin on a halfword boundary. The first two bytes contain a binary count of the number of bytes in the list (excluding the count field). Each entry in the list must occupy an 8-byte field. The sequence of entries is:

Entry	Standard DD name	Entry	Standard DD name	Entry	Standard DD name
1	not applicable	3	not applicable	5	SYSIN
2	not applicable	4	not applicable	6	SYSPRINT
				7	SYSPUNCH

If you omit an applicable entry, the translator uses the standard DD name. If you use a DD name less than 8 bytes long, fill the field with blanks on the right. You can omit an entry by placing X'FF' in the first byte. You can omit entries at the end of the list entirely.

The CICS-supplied interface modules

Each of your application programs to run under CICS requires one or more interface modules (also known as **stubs**) to use the following facilities:

- The EXEC interface
- The CPI Communications facility
- The SAA Resource Recovery facility
- The CICSPlex SM application programming interface (for information about CICSPlex SM stubs, see *CICSPlex SM Application Programming Guide*).

The interface modules and their use is described in the following sections.

The EXEC interface modules

Each of your CICS application programs must contain an interface to CICS. This takes the form of an EXEC interface module, used by the CICS high-level programming interface. The module, installed in the CICSTS13.CICS.SDFHLOAD library, must be link-edited with your application program to provide communication between your code and the EXEC interface program, DFHEIP.

The following interface modules are required for the assembler, C, COBOL, and PL/I programming languages:

Table 8. Language and Interface module name

Language	Interface module name
Assembler	DFHEAI and DFHEA10
C	DFHELII
COBOL	DFHECI
PL/I	DFHPL1OI supplied by PL/I (and DFHEPI, which is part of the PL/I DFHPL1OI module)
LE conforming	DFHELII

The CPI Communications interface module

Each of your CICS application programs that uses the Common Programming Interface for Communications (CPI Communications) must contain an interface to CPI Communications. This takes the form of an interface module, used by the CICS high-level programming interface, common to all the programming languages. The module, DFHCPLC, installed in the CICSTS13.CICS.SDFHLOAD library, must be link-edited with each application program that uses CPI Communications.

The SAA Resource Recovery interface module

Each of your CICS application programs that uses SAA Resource Recovery must contain an interface to SAA Resource Recovery. This takes the form of an interface module, used by the CICS high-level programming interface, common to all the programming languages. The module, DFHCPLRR, installed in the CICSTS13.CICS.SDFHLOAD library, must be link-edited with each application program that uses the SAA Resource Recovery facility.

Adding support for programming languages

This section describes the steps necessary to add support for the programming languages that can be used with CICS. You should normally complete appropriate actions described in the following sections before installing your application programs.

To write CICS application programs that request CICS services through the command-level application programming interface (API), you can use assembler language, C and C++, COBOL, or PL/I. You can also write application programs using C++ and Java, using the CICS OO foundation classes for C++, and the JCICS classes for Java.

CICS provides the support needed to run application programs written in assembler language, and OS/390 Language Environment (LE) provides the required support for all the other languages. The support provided by OS/390 LE covers:

- Programs compiled by the Language Environment-conforming compilers:
  - IBM COBOL for MVS & VM (5688–197)
  - IBM PL/I for MVS & VM (5688–235)
  - IBM C/C++ for MVS (5655–121)
  - SAA AD/Cycle COBOL/370 (5688–197)
  - SAA AD/Cycle PL/I (5688–235)
  - SAA AD/Cycle C/370 (5688–216)

- Programs compiled by the older, non-LE-conforming, compilers:
  - OS PL/I Optimizing Compiler Version 2 Release 1 (5668–910)
  - OS PL/I Optimizing Compiler Version 1 Release 5 (5724–PL1)
  - VS COBOL II (5668–958 and 5668–023)
  - OS/VS COBOL
  - C/370 (5688–040 and 5688–187)

If, for some reason, you choose not to use OS/390 LE, the alternative is to install runtime support in CICS for each of the old compilers used to compile your application programs (VS COBOL II, OS/VS COBOL, PL/I, and C). However, this is not recommended

Installing OS/390 LE support for CICS is discussed under “OS/390 LE support”.

Installing runtime support for the old compilers is discussed under “Native language support for non-LE compilers” on page 30.

OS/390 LE support

This section describes CICS support for OS/390 LE and what to do to install that support.

OS/390 LE support is provided by run-time libraries that establish a common execution environment for application programs compiled by high-level languages. You are recommended to run all programs compiled by a high-level language, whether by an LE-conforming compiler or not, under CICS-LE support.

The CICS-LE interface is initialized automatically if CICS can:

1. Load the LE interface modules, CEECCICS, CEEPIPI, and CEECTCB, from STEPLIB.
2. Successfully call the CEECCICS module to initialize the interface.

LE initialization takes place during CICS startup, when CICS issues message `DFHAP1203I` *`applid`* `Language Environment/370 is being initialized`. The CEECCICS module is loaded, followed by a partition initialization call to it, before the start of second phase PLT processing. If LE cannot successfully complete the initialization of all languages supported by CICS, or can only initialize some of them, it issues messages to the MVS console. If LE initialization fails completely, it may be because the CEECCICS module could not be loaded, or something went wrong during the loading of a particular language routine.

Installing CICS support for Language Environment

To enable OS/390 LE support to be installed correctly by CICS:

- Specify enough storage for the ERDSA to run CICS and Language Environment together. They need a minimum of 3500KB. To this minimum, add an amount of storage sufficient for your own requirements.
- Ensure the CICS-LE interface module, CEECCICS, and the LE modules CEEPIPI and CEECTCB are installed in an APF-authorized library defined in the STEPLIB concatenation in the CICS startup JCL. For example, include the LE SCEERUN library in the CICS STEPLIB, first making sure it is APF-authorized.

- Add the program resource definitions for the LE language interface modules to the CICS CSD. These are supplied as DEFINE statements in the CEECCSD member of the SCEESAMP library. After you have defined the program resource definitions, add the resource group to a CICS startup group list named in the GRPLIST system initialization parameter.

  Alternatively, CICS can create and install the resource definitions dynamically using program autoinstall. For more information about installing program resource definitions, see "Defining programs, map sets, and partition sets to CICS" on page 56.
- Define the LE transient data destinations, CESE, and CESO (DD names CEEMSG and CEEOUT). The CICS-supplied resource definition group, in the CSD, DFHDCTG, contains entries for CESE and CESO.

  For information about the attributes needed for LE transient data destinations, see the *IBM Language Environment for MVS & VM Programming Guide*, SC26-4818.
- Define the OS/390 LE runtime libraries on the CICS STEPLIB and DFHRPL DD statements as follows:
  - Add the SCEERUN library, which contains CEECCICS and CEECTCB, to STEPLIB.
  - Add the SCEECICS and SCEERUN libraries to DFHRPL, with SCEECICS concatenated *before* the SCEERUN library.

  For example:

```
//*               CICS APF-authorized libraries
//STEPLIB  DD DSN=hlq.CICS.SDFHAUTH,DISP=SHR
//         DD DSN=hlq.LE.SCEERUN,DISP=SHR
//*               CICS load libraries
//DFHRPL   DD DSN=hlq.CICS.SDFHLOAD,DISP=SHR
//         DD DSN=hlq.LE.SCEECICS,DISP=SHR
//         DD DSN=hlq.LE.SCEERUN,DISP=SHR
```

  Use only these LE runtime libraries for *all* your high-level language application programs, including those compiled with old, non-LE-conforming compilers, such as VS COBOL II and OS/VS COBOL.

Language Environment support for COBOL

LE is a prerequisite for application programs compiled using IBM COBOL for MVS and VM and SAA AD/Cysle COBOL/370. OS/390 LE incorporates the run-time libraries required for both these COBOL compilers. For information about OS/390 LE, see the *OS/390 Language Environment Customization* manual, SC28-1941.

To run under CICS your COBOL application programs compiled by an LE-conforming compiler:

- Install support for LE, ensuring CICS can initialize the LE enviroment during startup.
- Install resource definitions for your programs with the LANGUAGE attribute specified as LANGUAGE(LE370), or leave the language blank.

For your application programs, CICS can create and install program resource definitions automatically or you can create them specifically in the CSD, and install them by using the GRPLIST system initialization parameter or CEDA INSTALL command. For more information about installing program resource definitions, see "Defining programs, map sets, and partition sets to CICS" on page 56.

Run your VS COBOL II and OS/VS programs under LE using only the OS/390 LE runtime libraries. If the CICS-LE interface is not enabled, your COBOL application programs complied by non-LE-conforming compilers require the runtime support provided by their respective compilers.

For information about Language Environment support for programming languages, see the *Program Directory for IBM Language Environment for MVS and VM*.

Language Environment support for C and C++

LE is a prerequisite for application programs compiled using IBM C/C++ for MVS or SAA AD/Cycle C/370 compilers. OS/390 LE incorporates the run-time libraries required for both these C language compilers. For information about OS/390 LE, see the *OS/390 Language Environment Customization* manual, SC28-1941.

To run under CICS your C application programs compiled by an LE-conforming compiler:

- Install support for LE, ensuring CICS can initialize the LE enviroment during startup.
- Install resource definitions for your programs with the LANGUAGE attribute specified as LANGUAGE(C) or LANGUAGE(LE370), or leave the language blank.

For information about installing program resource definitions, see “Defining programs, map sets, and partition sets to CICS” on page 56.

CICS supports application programs written in C++ that:

- Are compiled using the IBM C/C++ for MVS compiler (5655-121)
- Execute with the OS/390 LE run-time libraries

Specify LANGUAGE(LE370) if:

- The program exploits Language Environment multi-language support.
- You intend to use LE support at run time. (If you do not intend to use Language Environment support at run time, code LANGUAGE(C).)
- The program is a C++ program.
- The program has been compiled by an LE-conforming compiler (IBM C/C++ for MVS (5655–121) or IBM SAA AD/Cycle C/370).

If you use Version 3 Release 2, or later, of the C/C++ compiler to compile a C++ program, specify the CXX parameter when options are passed to the compiler, otherwise the C compiler is invoked. Don’t specify CXX if a C program is to be compiled. See the *IBM C/C++ for MVS/ESA Compiler and Run-Time Migration Guide Version 3 Release 2*, SC33-2002, for further information.

For information about Language Environment support for programming languages, see the *Program Directory for IBM Language Environment for MVS and VM*.

Language Environment support for PL/I

LE is a prerequisite for application programs compiled using IBM PL/I for MVS or SAA AD/Cycle PL/I compilers. OS/390 LE incorporates the run-time libraries required for both these PL/I compilers. For information about OS/390 LE, see the *OS/390 Language Environment Customization* manual, SC28-1941.

To run CICS PL/I application programs compiled by an LE-conforming compiler:

- Install support for LE, ensuring CICS can initialize the LE enviroment during startup.
- Install resource definitions for the programs with the LANGUAGE attribute specified as LANGUAGE(PLI) or LANGUAGE(LE370), or leave blank.

For information about installing program resource definitions, see “Defining programs, map sets, and partition sets to CICS” on page 56.

Code LANGUAGE(LE370) if:

- The program exploits Language Environment multi-language support.
- You intend to use Language Environment support at run time. (If you don’t intend to use Language Environment support at run time, code LANGUAGE(PLI).)
- The program has been compiled by an LE-conforming compiler (IBM PL/I for MVS (5688–235) or IBM SAA AD/Cycle PL/I).

For information about LE support for programming languages, see the *Program Directory for IBM Language Environment for MVS and VM*.

Language Environment support for CICS JVM programs

LE is a prerequisite for CICS JVM programs. Unlike the other languages, JVM programs do not require the CICS-LE interface. JVM programs run with LE support using MVS services (not CICS services), which means that you do *not* need the LE runtime libraries defined on the DFHRPL DD statement. JVM programs require the LE support provided by the SCEERUN library only, which can be defined either in the CICS STEPLIB, or included in the MVS linklist.

Native language support for non-LE compilers

To run your high-level language application programs without the support of OS/390 LE runtime libraries requires the runtime support provided by the individual language. The runtime support required for each of the old non-LE-conforming compilers, if you choose not to use LE, is discussed in this section. It covers:

- “Installing CICS support for VS COBOL II”
- “Installing CICS support for C/370” on page 32
- “Installing CICS run-time support for PL/I” on page 32

Installing CICS support for VS COBOL II

If you choose not to use OS/390 LE support for your application programs written in VS COBOL II, CICS requires runtime support for VS COBOL II, installed as follows:

1. Place the following four VS COBOL II library routines in an APF-authorized library in the STEPLIB concatenation of your CICS startup job (for example, in the CICSTS13.CICS.SDFHAUTH library), or in an APF-authorized library in the MVS LNKLSTnn concatenation:
   a. IGZ9CIC (and its alias IGZECIC)
   b. IGZ9WTO (and its alias IGZEWTO)
   c. IGZ9OPD (and its alias IGZEOPD)
   d. IGZCMTxx, where xx represents the first two letters of the language specified on the MVS LANGUAGE option for your site (for example, IGZCMTEN)

After installing VS COBOL II, the IGZ9CIC, IGZ9WTO, and IGZ9OPD routines are located in the SYS1.COB2CICS library and the IGZCMTxx routine is located in the SYS1.COB2LIB library.

Alternatively, SYS1.COB2CICS can be APF-authorized and placed before the SYS1.COB2LIB library in the STEPLIB or JOBLIB.

Note: To use VS COBOL II, you need the CICS-VS COBOL II interface module, IGZECIC, in an APF-authorized library in the CICS STEPLIB concatenation. Do not put it in the LPA, because the LPA is not searched for this module.

2. Include the libraries containing the VS COBOL II library routines in the DFHRPL concatenation of your CICS startup JCL. VS COBOL II requires two packages of subroutines, known as COBPACKs. These subroutines are in two categories: (1) general and (2) environment-specific, containing system-specific logic. The COBPACKs you need are:

   IGZCPCC
   This module contains the CICS environment-specific modules, and is supplied in the SYS1.COB2CICS library.

   IGZCPAC
   This module contains the general VS COBOL II subroutines, and is supplied in the SYS1.COB2LIB library.

   Ensure that the SYS1.COB2CICS library is in front of the SYS1.COB2LIB library in the DFHRPL concatenation.

3. Create and install program resource definitions for the COBPACKs, with the LANGUAGE(ASSEMBLER) attribute. CICS can create and install program resource definitions automatically or you can create them specifically in the CSD, and install them by using the GRPLIST system initialization parameter or CEDA INSTALL command. For more information about installing program resource definitions, see “Defining programs, map sets, and partition sets to CICS” on page 56.

   The IGZECIC module does *not* have to be defined to CICS as a program resource in the CSD. For any other VS COBOL II library routine modules not included in COBPACKs, CICS requires program resource definitions, created either through program autoinstall automatically or explicitly defined.

If you decide to use program autoinstall to install the VS COBOL II COBPACKs resource definitions, remember to specify ACTIVE on the PGAIPGM system initialization parameter (the default is INACTIVE). The PGAIPGM system initialization parameter is described on page 269.

If you choose to define the VS COBOL II COBPACKs specifically, you can use the following commands:

```
DEFINE PROGRAM(IGZCPCC) GROUP(cob2grp) LANGUAGE(ASSEMBLER) CEDF(NO)
DEFINE PROGRAM(IGZCPAC) GROUP(cob2grp) LANGUAGE(ASSEMBLER) CEDF(NO)
ADD GROUP(cob2grp) LIST(listname)
```

For information about installing VS COBOL II support for CICS and about running VS COBOL II applications with CICS, see the *VS COBOL II Installation and Customization* manual.

OS/VS COBOL and storage protection: To run OS/VS COBOL programs in a CICS environment that has storage protection active, CICS loads the reentrant

OS/VS COBOL compatibility modules (those whose names begin with ILB) from an APF-authorized library in the CICS STEPLIB. Include the OS/VS runtime libarary containing these modules in the CICS STEPLIB, making sure it is APF-authorized.

CICS loader also loads ILBOCOM from the DFHRPL concatenation. Ensure the module is available from the OS/VS runtime library in the CICS DFHRPL library.

Note:

Installing CICS support for C/370

If you choose not to use OS/390 LE support for your application programs written in C, CICS requires C/370 runtime support as follows:

1. Install the C/370 library and compiler. For information about this, see the *IBM C/370 Installation Guide*). When you have installed C/370, the SEDCLINK library contains the two load modules, EDCCICS and EDCXV, required for CICS-C/370 support.
2. Generate CICS support for C/370, as follows:
   a. Copy the CICS-C/370 interface module, EDCCICS, from SEDCLINK to one of the APF-authorized libraries defined in the STEPLIB DD statement in the CICS startup JCL.
   b. Create and install a resource definition for the EDCXV module with the LANGUAGE(ASSEMBLER) attribute.

      CICS can create and install program resource definitions automatically or you can create them specifically in the CSD, and install them by using the GRPLIST system initialization parameter or CEDA INSTALL command. For more information about installing program resource definitions, see “Defining programs, map sets, and partition sets to CICS” on page 56.

      The following commands are examples of defining the module and adding the group in which it is defined to the list, INSTLIST, to be installed at CICS initialization:

      ```
      DEFINE PROGRAM(EDCXV) GROUP(C370) LANGUAGE(ASSEMBLER)
      ADD GROUP(C370) LIST(INSTLIST)
      ```

   c. Define the C/370 run-time library, SEDCLINK, in the DFHRPL statement in the CICS startup job stream.
   d. Create and install resource definitions for the C/370-provided locales in the same way as for EDCXV. These locales are EDC$GERM, EDC$USA, EDC$FRAN, EDC$ITAL, and EDC$SPAI. These are all load modules in the SEDCLINK library.

      Note: Locale is the term defined by the American National Standard for Information Systems (ANSI) to denote a C/370 programming language environment for a given national language. The C/370-supplied locales provide a C/370 programming language environment for German (EDC$GERM), American English (EDC$USA), French (EDC$FRAN), Italian (EDC$ITAL) and Spanish (EDC$SPAI).

For information about C/370, see the *IBM C/370 Programming Guide*.

Installing CICS run-time support for PL/I

If you choose not to use OS/390 LE support for your application programs written in PL/I, CICS requires PL/I runtime support in the form of resource definitions and runtime libraries. For information about the definitions required, see the *OS PL/I*

Version 2 Installation and Customization under MVS Guide. For information about installing program resource definitions, see “Defining programs, map sets, and partition sets to CICS” on page 56.

Note: CICS run-time support for PL/I Version 2.3 also supports programs compiled against earlier releases of PL/I.

The group of CSD definitions supplied by CICS in earlier releases, DFHPLI, is not supplied in CICS TS, nor is it supplied in one of the compatibility groups. For an explanation about compatibility groups in general, see Sharing the CSD between different releases of CICS and “CICS-supplied compatibility groups” on page 146. (The *CICS Resource Definition Guide* lists the contents of each compatibility group.)

Generating PL/I shared library support for CICS

If you want your PL/I application programs to run with the PL/I shared library facility, ensure that you generate the PL/I shared library modules. To do this run the stage 1 and stage 2 generation jobs described in the *OS PL/I Version 2 Installation and Customization under MVS Guide*. The stage 1 job assembles the PL/I macro, PLRSHR, supplied in the SYS1.SHRMAC library. The assembly of the PLRSHR macro generates the stage 2 jobs to assemble and link-edit the following modules:

- PLISHRE
- IBMBPSLA
- IBMBPSMA
- IBMBPSRA
- IBMTPSLA
- IBMTPSRA

When you have completed the stage 2 jobs, which link-edit the PL/I shared library modules into the SYS1.PLIBASE library (or another suitable library), ensure that modules IBMBPSLA and IBMBPSMA are also installed in one of the libraries in the CICS DFHRPL library concatenation (for example, the CICSTS13.CICS.SDFHLOAD library) or in the LPA.

When you start up CICS, it attempts to load the modules IBMBPSLA and IBMBPSMA into the CICS nucleus. If this load fails (for example, because the modules are not found), PL/I shared library support is not available.

Also, run the job shown in Figure 11, to link-edit module PLISHRE into the CICSTS13.CICS.SDFHLOAD library.

```
//PLISHRE  JOB  'accounting information',CLASS=A,MSGCLASS=A
//LNKEDIT  EXEC DFHLNKVS,NAME=LOADLIB,INDEX=CICSTS13.CICS,
//              INDEX2=CICSTS13.CICS
//SYSPUNCH DD DUMMY
//SYSLIB   DD DSN=SYS1.PLIBASE,DISP=SHR
//         DD DSN=SYS1.SIBMBASE,DISP=SHR
//SYSLIN   DD *
           REPLACE IBMBPIR1
           INCLUDE SYSLIB(PLISHRE)
           NAME    PLISHRE(R)
/*
//
```

Figure 11. Job to link-edit PLISHRE

Preparing to install application programs

This section describes the following points that you should consider about application programs to be installed and the libraries that they are to be installed into:

- Using BMS map sets in programs
- MVS residence and addressing mode
- Program eligibility for the MVS link pack area (LPA)
- Installing programs in load library secondary extents

This section also outlines the steps to install application programs to run under CICS. (See “Overview of installing application programs” on page 40.)

Using BMS map sets in application programs

This section describes what to do to use BMS map sets in application programs.

Before you install an application program to run under CICS:

- Create any BMS map sets used by the program, as described in “Chapter 3. Installing map sets and partition sets” on page 13.
- Include the physical map sets (used by BMS in its formatting activities) in a library that is in the DFHRPL concatenation.
- Either include the symbolic map sets (copied into the application programs) in a user copy library or insert them directly into the application program source.

The DFHMAPS procedure writes the symbolic map set output to the library specified on the DSCTLIB parameter, which defaults to the CICSTS13.CICS.SDFHMAC library. If you want to include symbolic map sets in a user copy library:

- Specify the library name by the *DSCTLIB=name* operand on the EXEC statement for the DFHMAPS procedure used to install physical and symbolic map sets together.
- Include a DD statement for the user copy library in the SYSLIB concatenation of the job stream used to assemble and compile the application program.

  If you choose to let the DFHMAPS procedure write the symbolic map sets to the CICSTS13.CICS.SDFHMAC library (the default), include a DD statement for the CICSTS13.CICS.SDFHMAC library in the SYSLIB concatenation of the job stream used to compile the application program. This is not necessary for the DFHEITAL procedure used to assemble assembler-language programs, because these jobs already include a DD statement for the CICSTS13.CICS.SDFHMAC library in the SYSLIB concatenation.
- For PL/I, specify a library that has a block size of 400 bytes. This is necessary to overcome the blocksize restriction on the PL/I compiler.

For more information about installing map sets, see “Chapter 3. Installing map sets and partition sets” on page 13. For information about writing programs to use BMS services, see the *CICS Application Programming Guide*.

MVS residence and addressing modes

This section describes the effect of the MVS residence and addressing modes on application programs, how you can change the modes, and how you can make application programs permanently resident. An application written to run on MVS/370 can run on any MVS system, if it is link-edited with the AMODE(24) and RMODE(24) options.

A command-level program written in assembler language, C/370, VS COBOL II, or PL/I Version 1 Release 5.1 or later, can reside above 16MB, and address areas above 16MB. The program can contain EXEC CICS, EXEC DLI, and CALL DL/I commands.

Establishing a program's addressing mode

Every program that executes in MVS is assigned two attributes: an addressing mode (AMODE), and a residency mode (RMODE). AMODE specifies the addressing mode in which your program is designed to receive control. Generally, your program is designed to execute in that mode, although you can switch modes in the program, and have different AMODE attributes for different entry points within a load module. The RMODE attribute indicates where in virtual storage your program can reside. Valid AMODE and RMODE specifications are:

AMODE(24)	Specifies 24-bit addressing mode.
AMODE(31)	Specifies 31-bit addressing mode.
AMODE(ANY)	Specifies either 24- or 31-bit addressing mode.
RMODE(24)	Indicates that the module must reside in virtual storage below 16MB. You can specify RMODE(24) for 31-bit programs that have 24-bit dependencies.
RMODE(ANY)	Indicates that the module can reside anywhere in virtual storage.

Note: C/370 language programs must be link-edited with AMODE(31).

If you do not specify any AMODE or RMODE attributes for your program, MVS assigns the system defaults AMODE(24) and RMODE(24). To override these defaults, you can specify AMODE and RMODE in one or more of the following places. Assignments in this list overwrite assignments later in the list.

1. On the linkage editor MODE control statement:
   ```
   MODE AMODE(31),RMODE(ANY)
   ```
2. Either of the following:
   a. In the PARM string on the EXEC statement of the linkage editor job step:
      ```
      //LKED    EXEC  PGM=IEWL,PARM='AMODE(31),RMODE(ANY),..'
      ```
   b. On the LINK TSO command, which causes processing equivalent to that of the EXEC statement in the linkage editor step.
3. On AMODE or RMODE statements within the source code of an assembler program. (You can also set these modes in COBOL by means of the compiler options; for information about COBOL compiler options, see the relevant application programming guide for your COBOL compiler.)
4. The link-edit modules DFHECI and DFHEPI assign AMODE(31) and RMODE(ANY) to COBOL and PL/I programs.

For information about these modes and the rules that govern their use, see the *DFSMS/MVS Program Management*.

CICS address space considerations

Table 9 gives the valid combinations of the AMODE and RMODE attributes and their effects:

Table 9. Valid AMODE and RMODE specifications and their effects

AMODE	RMODE	Residence	Addressing
24	24	Below 16MB	24-bit mode
31	24	Below 16MB	31-bit mode
ANY	24	Below 16MB	31-bit mode
31	ANY	Above 16MB	31-bit mode

The following example shows linkage editor control statements for a program coded to 31-bit standards:

```
//LKED.SYSIN DD *
   MODE AMODE(31),RMODE(ANY)
   NAME    anyname(R)    ("anyname" is your load module name)
/*
//
```

Making programs permanently resident

If you define a program in the CSD with the resident attribute, RESIDENT(YES), it is loaded on first reference. This applies to programs link-edited with either RMODE(ANY) or RMODE(24). However, be aware that the storage compression algorithm that CICS uses does not remove resident programs.

If there is not enough storage for a task to load a program, the task is suspended until enough storage becomes available. If any of the DSAs get close to being short on storage, CICS frees the storage occupied by programs that are not in use. (For more information about the dynamic storage areas in CICS, see “Storage protection” on page 353.)

Instead of making RMODE(24) programs resident, you can make them non-resident and use the library lookaside (LLA) function. The space occupied by such a program is freed when its usage count reaches zero, making more virtual storage available. LLA keeps its library directory in storage and stages (places) copies of LLA-managed library modules in a data space managed by the virtual lookaside facility (VLF). CICS locates a program module from LLA’s directory in storage, rather than searching program directories on DASD. When CICS requests a staged module, LLA gets it from storage without any I/O.

Preparing applications to run in the link pack area

Programs written in assembler language, C, VS COBOL II or later, or PL/I Release 5.1, can reside in the link pack area (LPA). To do so, they must be read-only and have been link-edited with the RENT and REFR options. Other requirements are as follows:

Assembler
Use the RENT assembler option.

C Use the RENT compiler option.

COBOL
Do not overwrite WORKING STORAGE. (The CICS translator generates a CBL

statement with the required compiler RENT option (unless you specify the translator option NOCBLCARD). For the COBOL and COBOL2 translator options, the translator also generates the RES compiler option.)

PL/I Version 1 Release 5.1 or later
Do not overwrite STATIC storage. (The CICS translator inserts the required REENTRANT option into the PROCEDURE statement.)

If you want CICS to use modules that you have written to these standards, and installed in the LPA, specify USELPACOPY(YES) on the program resource definitions in the CSD.

For information about installing CICS modules in the LPA, see the *CICS Transaction Server for OS/390 Installation Guide*.

Preparing application programs to run in the RDSAs

Programs that are eligible to reside above 16MB, and are read-only, can reside in the CICS extended read-only DSA (ERDSA). Therefore, to be eligible for the ERDSA, programs must be:

- Properly written to read-only standards
- Written to 31-bit addressing standards
- Link-edited with the RENT attribute and the RMODE(ANY) residency attribute

Note: OS/VS COBOL or pre-CICS/VS 1.6 (DFHE type program stub) programs are not re-entrant and therefore cannot be loaded into read-only storage.

If the default RENTPGM=PROTECT is specified as a system initialization parameter, these programs should be link-edited with the NORENT option and not RENT or REFR.

If RENTPGM=NOPROTECT is specified as a system initialization parameter, they can be link-edited using the RENT option. However, in this case they will be loaded into the CDSA or ECDSA.

Programs that are *not* eligible to reside above 16MB, and are read-only, can reside in the CICS read-only DSA (RDSA) below 16MB. Therefore, to be eligible for the RDSA, programs must be:

- Properly written to read-only standards
- Link-edited with the RENT attribute

Note: When you are running CICS with RENTPGM=PROTECT specified as a system initialization parameter, the RDSAs are allocated from key-0 read-only storage.

Programs link-edited with RENT and RMODE(ANY) are automatically loaded by CICS into the ERDSA.

ERDSA requirements for the specific languages are described in the following sections.

Assembler

If you want CICS to load your assembler programs in the ERDSA, assemble and link-edit them with the following options:

1. The RENT assembler option
2. The linkage-editor RENT attribute
3. The RMODE(ANY) residency mode

Note: If you specify these options, ensure that the program is truly read-only (that is, does not modify itself in any way—for example, by writing to static storage), otherwise storage exceptions occur. The program must also be written to 31-bit addressing standards. See the *CICS Problem Determination Guide* for some possible causes of storage protection exceptions in programs resident in the ERDSA.

The CICS-supplied procedure, DFHEITAL, has a LNKPARM parameter that specifies the XREF and LIST options only. To link-edit an ERDSA-eligible program, override LNKPARM from the calling job, specifying the RENT and RMODE(ANY) options in addition to any others you require. For example:

```
//ASMPROG  JOB 1,user_name,MSGCLASS=A,CLASS=A,NOTIFY=userid
//EITAL    EXEC DFHEITAL,
           .
           (other parameters as necessary)
           .
//         LNKPARM='LIST,XREF,RMODE(ANY),RENT'
```

Note: The CICS EXEC interface module for assembler programs (DFHEAI) specifies AMODE(ANY) and RMODE(ANY). However, because the assembler defaults your application to AMODE(24) and RMODE(24), the resulting load module also becomes AMODE(24) and RMODE(24).

If you want your application program link-edited as AMODE(31) and RMODE(ANY), you are recommended to use appropriate statements in your assembler program. For example:

```
MYPROG    CSECT
MYPROG    AMODE 31
MYPROG    RMODE ANY
```

There are two alternative methods of setting AMODE and RMODE:

- You can set the required AMODE and RMODE specification using linkage editor (or binder) control information in the JCL PARM string. For example:

  ```
  //EITAL  EXEC DFHEITAL,
           LNKPARM='LIST,XREF,RENT,AMODE(31),RMODE(ANY)'
  ```

- You can use the MODE control statement in the SYSIN dataset in the linkage editor or binder step in your JCL. However, when using the binder, you may see unexpected warning messages with regard to conflicting AMODE and RMODE specifications.

C and C/++

If you want CICS to load your C and C++ programs into the ERDSA, compile and link-edit them with:

1. The RENT compiler option.

The CICS-supplied procedures DFHYITDL (for C) and DFHYITEL (for C++) have a LNKPARM parameter that specifies a number of link-edit options. To link-edit an ERDSA-eligible program, override this parameter from the calling job, and add RENT to the other options you require. You do not need to add the RMODE(ANY) option, because the CICS EXEC interface module for C/370 (DFHELII) is link-edited

with AMODE(31) and RMODE(ANY). Therefore, your program is link-edited as AMODE(31) and RMODE(ANY) automatically when you include the CICS EXEC interface stub.

The following sample job statements show the LNKPARM parameter with the RENT option added:

```
//C370PROG   JOB 1,user_name,MSGCLASS=A,CLASS=A,NOTIFY=userid
//EYTDL    EXEC DFHEYTDL,
           .
           (other parameters as necessary)
           .
//         LNKPARM='LIST,MAP,LET,XREF,RENT'
```

COBOL

LE-conforming COBOL and VS COBOL II programs are automatically eligible for the ERDSA, because:

- If you use the translator option, CBLCARD (the default), the required compiler option, RENT, is included automatically on the CBL statement generated by the CICS translator. If you use the translator option, NOCBLCARD, specify the RENT option either on the PARM statement of the compile job step, or by using the COBOL macro IGYCOPT to set installation-defined options.
- The COBOL compiler automatically generates code that conforms to read-only and 31-bit addressing standards.
- The CICS EXEC interface module for COBOL (DFHECI) is link-edited with AMODE(31) and RMODE(ANY). Therefore, your program is link-edited as AMODE(31) and RMODE(ANY) automatically when you include the CICS EXEC interface stub.

You also need to specify the reentrant attribute to the linkage-editor. The CICS-supplied procedure, DFHYITVL (and also DFHEITVL), has a LNKPARM parameter that specifies a number of link-edit options. To link-edit an ERDSA-eligible program, override this parameter from the calling job, and add RENT to any other options you require. For example:

```
//COB2PROG JOB 1,user_name,MSGCLASS=A,CLASS=A,NOTIFY=userid
//YITVL    EXEC DFHYITVL,
           .
           (other parameters as necessary)
           .
//         LNKPARM='LIST,XREF,RENT'
```

PL/I

CICS PL/I programs are generally eligible for the ERDSA, provided they do not modify static storage. The following requirements are enforced, either by CICS or PL/I:

- The required REENTRANT option is included automatically, by the CICS translator, on the PL/I PROCEDURE statement.
- The PL/I compiler automatically generates code that conforms to 31-bit addressing standards.
- The CICS EXEC interface module for PL/I (DFHEPI, which is part of the PL/I DFHPL1OI module) is link-edited with AMODE(31) and RMODE(ANY). Therefore, your program is link-edited as AMODE(31) and RMODE(ANY) automatically when you include the CICS EXEC interface stub.

You also need to specify the reentrant attribute to the linkage-editor. The CICS-supplied procedure, DFHYITPL (and DFHEITPL), has a LNKPARM parameter that specifies a number of link-edit options. To link-edit an ERDSA-eligible program, override this parameter from the calling job, and add RENT to any other options you require. For example:

```
//PLIPROG  JOB 1,user_name,MSGCLASS=A,CLASS=A,NOTIFY=userid
//YITPL    EXEC DFHYITPL,
           .
           (other parameters as necessary)
           .
//         LNKPARM='LIST,XREF,RENT'
```

Note: Do not specify the RENT attribute on the link-edit step unless you have ensured the program is truly read-only (and does not, for example, write to static storage), otherwise storage exceptions will occur. See the *CICS Problem Determination Guide* for some possible causes of storage protection exceptions in programs resident in the ERDSA.

Installing programs in load library secondary extents

CICS supports load library secondary extents that are created while CICS is executing. If you define libraries in the DFHRPL concatenation with primary and secondary extents, and secondary extents are added as a result of link-editing into the DFHRPL library while CICS is running, the CICS loader detects the occurrence, closes, and then reopens the library. This means that you can introduce new versions using the CEMT NEWCOPY command, even if the new copy of the program has caused a new library extent.

However, this can increase the search time when loading modules from the secondary extents. You should avoid using secondary extents if possible.

Overview of installing application programs

This section outlines the steps required to install application programs to run under CICS. For detailed information about using the CICS-supplied procedures to install application programs, see “Using the CICS-supplied procedures to install application programs” on page 41. To use your own JCL to install application programs, see “Using your own job streams” on page 53.

1. Translate the program source code, turning CICS commands into code understood by the compiler.

   Notes:

   a. For a program that does not use CICS commands and is only invoked by a running transaction (and never directly by CICS task initiation), no translator step is needed.

   b. CICS command-level programs that access DL/I services through either the DL/I CALL **or** EXEC DLI interfaces must also be translated. Applications that access IBM DATABASE 2 (DB2) services using the EXEC SQL interface need an additional precompilation step. For information about this step, see the *IBM DATABASE 2 Application Programming and SQL Guide*.
2. Compile or assemble the translator output to produce object code.
3. Pre-link any C and C++ programs.
4. Link-edit the object module to produce a load module, which you store in an application load library that is concatenated to the DFHRPL DD statement of the CICS startup job stream. Additional INCLUDE statements are required for

applications that access DB2 services using the EXEC SQL interface. For information about these extra statements, see the *IBM DATABASE 2 Application Programming and SQL Guide*.

5. Create resource definition entries, in the CSD, for any transactions that invokes the program.
6. Do one of the following:
   - If you are using program autoinstall, ensure that the autoinstall user-replaceable module can correctly install a resource definition for the program.
   - If you are not using program autoinstall, create a resource definition entry in the CSD for the program.

If you have macro-level programs from an earlier release of CICS, recode them as command-level programs. Furthermore, references to the CSA or to the TCA are not allowed. You can specify YES for the system initialization parameter DISMACP to cause CICS to disable any transaction whose program invokes a CICS macro or references the CSA or the TCA.

CICS provides a utility program, DFHMSCAN, to identify the macro-level programs used by your CICS applications. For information about using the DFHMSCAN utility to identify macro-level programs, see the *CICS Operations and Utilities Guide*.

Using the CICS-supplied procedures to install application programs

The following sections describe the use of CICS-supplied procedures to install application programs to run under CICS.

Common considerations when installing application programs

This section outlines points to consider when installing application programs.

- All the sample job streams in the following sections apply if your program source statements are imbedded in the text. If your program source is a member of a partitioned data set, replace the //TRN.SYSIN statement by a statement in one of the following forms:

  ```
  //TRN.SYSIN DD DSN=partition.dataset.name(programname),DISP=SHR
  //SYSIN DD DSN=partition.dataset.name(programname),DISP=SHR
  ```

- If you want your application program to use CPI Communications or SAA Resource Recovery, make the appropriate interface modules available to your program. For information about the CPI Communications interface module and the SAA Resource Recovery interface module, see page 26.
- If you want your application program to reside in the MVS link pack area (LPA), specify appropriate options when installing your program. Options appropriate to each language are given for the sample job streams in the following sections. For information on preparing programs to run in the link pack area (LPA), see page 36.

  For information on preparing programs to run in the read-only DSAs, see page 37.
- If you want your application program to use BMS maps, first prepare the map sets. For more information, see “Using BMS map sets in application programs” on page 34.

Using the CICS-supplied interface modules

The CICS-supplied procedures to install your online application programs in a CICS library specify the CICS library member that contains the INCLUDE statement for the appropriate language EXEC interface module. These library members are named DFHEILIx, where x is A for assembler, C for COBOL, D for C, or P for PL/I. For example, the DFHYITVL procedure uses the following statements:

```
//COPYLINK EXEC PGM=IEBGENER,COND=(7,LT,COB)
//SYSUT1   DD DSN=&INDEX..SDFHCOB(&STUB),DISP=SHR
//SYSUT2   DD DSN=&&COPYLINK,DISP=(NEW,PASS),
//            DCB=(LRECL=80,BLKSIZE=400,RECFM=FB),
//            UNIT=&WORK,SPACE=(400,(20,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DUMMY
   :
   :
//SYSLIN   DD DSN=&&COPYLINK,DISP=(OLD,DELETE)
//         DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD DDNAME=SYSIN
```

In this COBOL example, the symbolic parameter STUB defaults to DFHEILIC. The DFHEILIC member contains the statement INCLUDE SYSLIB(DFHECI).

In the DFHEITPL procedure for installing PL/I application programs, the SYS1.PL1BASE library is included in the SYSLIB concatenation. This is needed because the INCLUDE statement in the DFHEILIP module specifies INCLUDE SYSLIB(DFHPL1OI), where DFHPL1OI is the name of the PL/I interface module supplied by PL/I. For more information about the DFHPL1OI module, see the notes to Figure 16 on page 49.

In the DFHEILIP module, after the INCLUDE statement, there is the REPLACE PLISTART linkage editor command. This command causes the CSECT PLISTART, which is inserted by the compiler, to be removed because equivalent function is in the stub DFHPL1OI. The REPLACE PLISTART command is needed for programs to run under Language Environment, but is optional for other PL/I programs that run under CICS.

If your application program is to use CPI Communications or the SAA Resource Recovery facility, do one of the following:

- Add appropriate INCLUDE statements to the LKED.SYSIN override in the job used to call the CICS-supplied procedure to install your application program. Add the following INCLUDE statements:
  - INCLUDE SYSLIB(DFHCPLC) if your program uses CPI Communications
  - INCLUDE SYSLIB(DFHCPLRR) if your program uses SAA Resource Recovery
- Rely on the linkage editor automatic library-call mechanism to include the necessary modules.

For more information about linkage editor requirements, see “Using your own job streams” on page 53.

Installing assembler language application programs

You can use the DFHEITAL procedure (see Figure 12 on page 43) to translate, assemble, and link-edit application programs written in assembler language.

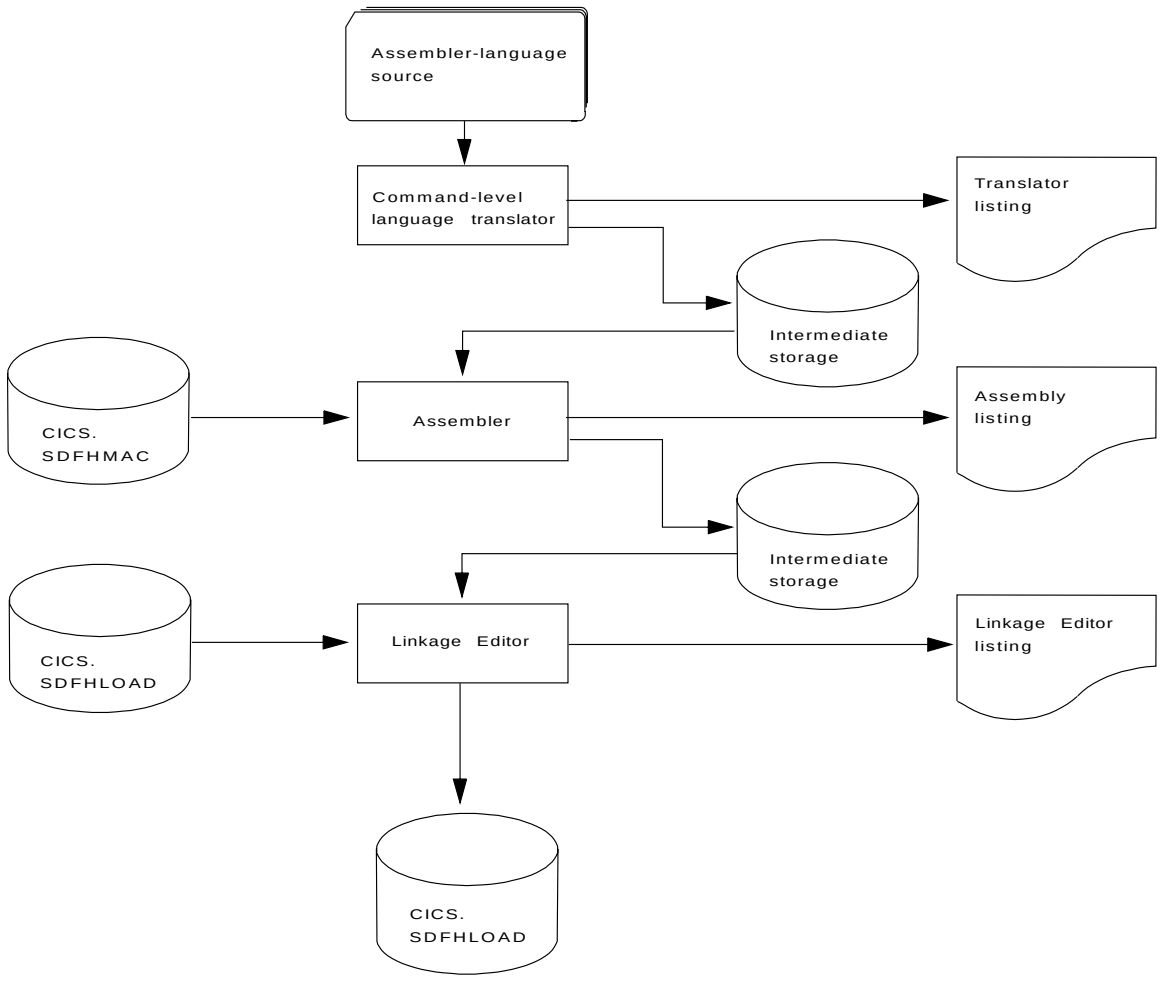


Figure 12. Installing assembler language programs using the DFHEITAL procedure

Sample JCL to install assembler application programs

You can use the sample job control statements shown in Figure 13 on page 44 to process application programs written in assembler language. In the procedure name, “x” depends on whether your programs are CICS application programs or EXCI batch programs. For the names of the CICS-supplied procedures, see Table 7 on page 24.

```
//jobname      JOB     accounting info,name,MSGLEVEL=1
//             EXEC    PROC=DFHExTAL                        1
//TRN.SYSIN    DD      *
*ASM      XOPTS(translator options . . .)                   2
               .
               assembler-language source statements
               .
/*
//LKED.SYSIN   DD      *
               NAME    anyname(R)
/*
//
```

where `anyname` is your load module name

Figure 13. Sample job control statements to call the DFHwxTAL procedures

Notes:

1 If you are installing a program into either of the read-only DSAs, see “Preparing application programs to run in the RDSAs” on page 37 for more details.

If you are installing a program that is to be used from the LPA, add:

- RENT to the PARM options in the EXEC statement for the ASM step of the DFHEITAL procedure
- RENT and REFR options to the LNKPARM parameter on the call to the DFHEITAL procedure

(See “Preparing applications to run in the link pack area” on page 36.)

2 For information about the translator options you can include on the XOPTS statement, see Specifying translator options in the *CICS Application Programming Guide* .

Installing COBOL application programs

Figure 14 on page 45 illustrates the flow of control in the cataloged procedures for COBOL and PL/I programs.

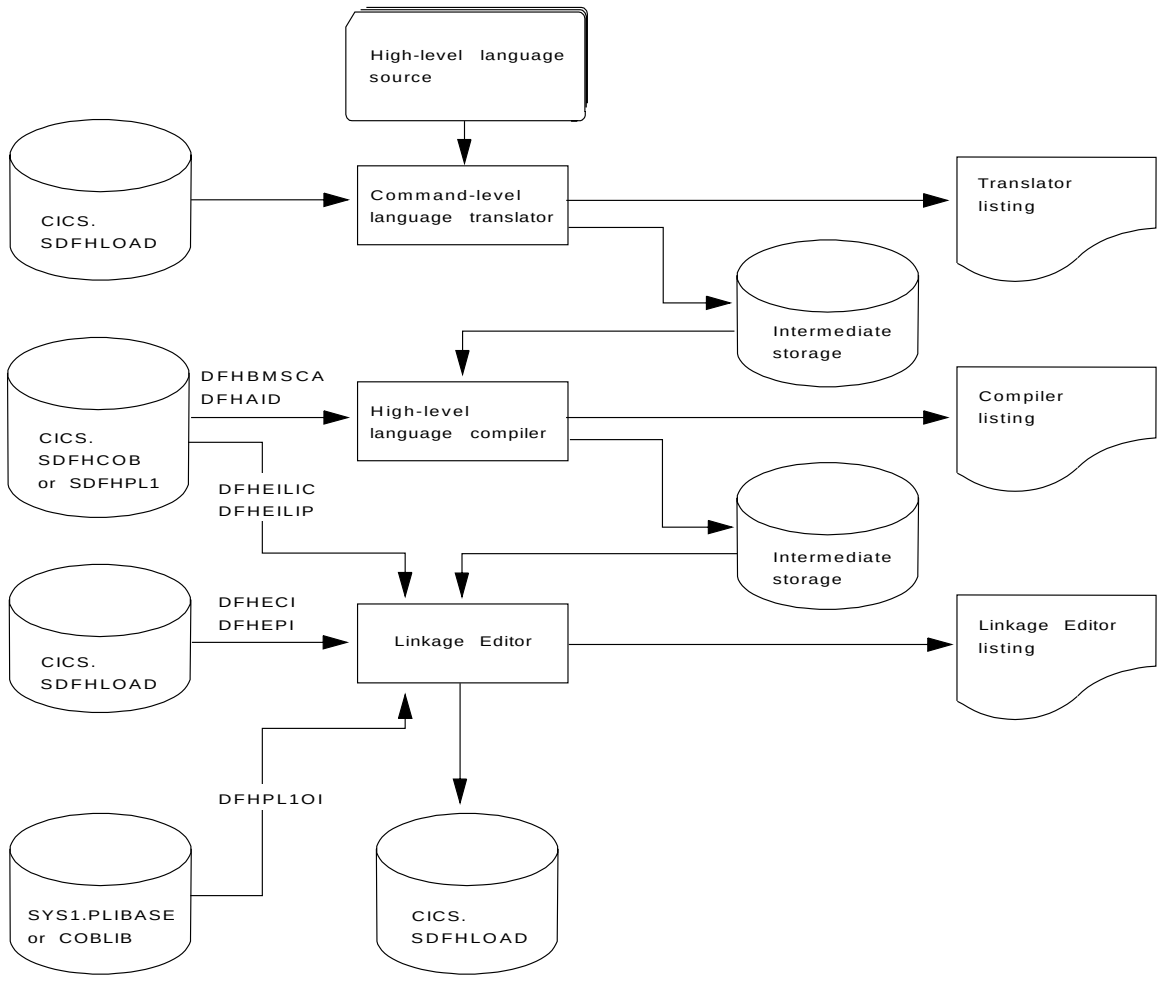


Figure 14. Installing COBOL and PL/I programs

Sample JCL to install COBOL application programs

You can use the job control statements shown in Figure 15 on page 46 to process COBOL application programs. In the procedure name, “wx” depends on whether your programs are to be compiled by an LE-conforming compiler or otherwise, and whether it is a CICS application program or an EXCI batch program. For the names of the CICS-supplied COBOL procedures, see Table 7 on page 24.

For information about adding CICS support for OS/390 LE, see “Installing CICS support for Language Environment” on page 27. For information about adding CICS support for VS COBOL II, see “Installing CICS support for VS COBOL II” on page 30.

```
//jobname    JOB  accounting info,name,MSGLEVEL=1
//           EXEC PROC=DFHwxTVL                          1
//TRN.SYSIN  DD      *                                   2
 CBL   XOPTS(Translator options . . .)                   3
             .
            COBOL source statements
             .
/*
//LKED.SYSIN DD      *                                   4
              NAME    anyname(R)
/*
//
```

where `anyname` is your load module name

Figure 15. Sample job control statements to call the DFHwxTVL procedures

Notes for installing COBOL programs

1 Translator options:

Specify the following translator options according to the version of the COBOL compiler invoked in the compile step.

OOCOBOL
for the IBM COBOL for MVS and VM compiler. This option is only needed if the object-oriented syntax (such as class-id and method-id), is used in the application program. The OOCOBOL option implies the COBOL3, ANSI85 and COBOL2 translator options.

COBOL3
for one of the LE-conforming COBOL compilers. COBOL3 implies the ANSI85 and COBOL2 translator options.

ANSI85
for the VS COBOL II compiler. This option specifies that the translator is to translate VS COBOL II programs that implement the ANSI85 standards. ANSI85 implies the COBOL2 option.

COBOL2
for the VS COBOL II compiler.

Compiler options:

To compile a VS COBOL II application program, you need the compiler options: RENT, RES, NODYNAM, and LIB. If you use the translator option, CBLCARD (the default), the CICS translator automatically generates a CBL statement containing these options. You can prevent the generation of a CBL or PROCESS card by specifying the translator option NOCBLCARD.

The PARM statement of the COB step in DFHwxTVL specifies values for the compiler options. For example,

```
//COB    EXEC PGM=IGYCRCTL,REGION=&REG,
//       PARM='NODYNAM,LIB,OBJECT,RENT,RES,APOST,MAP,XREF'
```

It does not specify values for the SIZE and BUF options. The defaults are SIZE=MAX, and BUF=4K. SIZE defines the amount of virtual storage available to the compiler, and BUF defines the amount of dynamic storage to be allocated to

each compiler buffer work file. You can change these options with a PARM.COB parameter in the EXEC statement that invokes the procedure. For example:

```
EXEC  PROC=DFHwxTVL,PARM.COB='SIZE=512K,BUF=16K,.,.,.'
```

Ensure that the APOST|QUOTE option in effect for the COBOL compiler matches that for the translator.

There is no BATCH compiler option for VS COBOL II. For information about VS COBOL II compiler options, see the *VS COBOL II Application Programming Guide*.

You can change compiler options by using any of the following methods:

- By overriding the PARM statement defined on the COB step of the DFHwxTCL procedure.

  If you specify a PARM statement in the job that calls the procedure, it overrides **all** the options specified in the procedure JCL. Ensure that all the options you want are specified in the override, or in a CBL statement.
- Specifying a CBL statement at the start of the source statements in the job stream used to call the DFHwxTCL procedure.
- The COBOL installation defaults macro, IGYCOPT.

  This is needed if you do not use a CBL statement; that is, have specified the translator option NOCBLCARD, and the compiler option ALOWCBL=NO.

For information about the translator option CBLCARD|NOCBLCARD, see the *CICS Application Programming Guide*. If you choose to use the NOCBLCARD option, also specify the COBOL compiler option ALOWCBL=NO to prevent an error message of IGYOS4006-E being issued. The ALOWCBL=NO option can be overridden at compile time by the JCL PARM option or a TSO command. For more information about the ALOWCBL compiler option, see the relevant *Installation and Customization* manual for your version of COBOL.

2 If you have no input for the translator, you can specify `DD DUMMY` instead of `DD *`. However, if you specify `DD DUMMY`, also code a suitable DCB operand. (The translator does not supply all the data control block information for the SYSIN data set.)

3 The translator options on the XOPTS statement override similar options in the DFHwxTVL procedure.

Any translator options you specify should include the type of COBOL translator option, COBOL2 or COBOL3.

For information about the translator options you can include on the XOPTS statement, see the Specifying translator options in the *CICS Application Programming Guide* .

4 You can ignore weak external references unresolved by the linkage editor.

The link-edit job step requires access to the libraries containing the environment-specific modules for CICS, the general VS COBOL II library subroutines, and the Language Environment link-edit modules, as appropriate. The required libraries are included in the SYSLIB concatenation of procedures for VS COBOL II. Override or change the names of these libraries if the modules and library subroutines are installed in libraries with different names.

If you are installing a program into either of the read-only DSAs, see “Preparing application programs to run in the RDSAs” on page 37 for more details.

If you are installing a program that is to be used from the LPA, add the RENT and REFR options to the LNKPARM parameter on the call to the DFHEITVL procedure. (See “Preparing applications to run in the link pack area” on page 36.)

Installing PL/I application programs

Figure 14 on page 45 illustrates the flow of control in the cataloged procedures for PL/I programs.

PL/I-CICS application programs must be link-edited in a different way than non-CICS PL/I applications. This is because the normal entry point, control section PLISTART, is not required on CICS systems. Instead, the module DFHPL1OI is provided by PL/I, which acts as the entry point to the program and must be link-edited with the application program. The CICS loader requires that DFHPL1OI be positioned at the head of the load module.

For example, assuming the PL/I application is made up of members PLIAPP and EXTSUB, which reside in the library defined by ddname L1, and that the SYSLIB data set contains DFHPL1OI, then the following linkage editor statements should be used:

```
INCLUDE  SYSLIB (DFHPL1OI)
      Ensure that DFHPL1OI is at the head of the load module.
REPLACE  PLISTART
      Delete unwanted CSECTs from following INCLUDE.
INCLUDE  L1 (PLIAPP)
      Application procedure (1).
REPLACE  PLISTART
      Delete unwanted CSECTs from following INCLUDE.
INCLUDE  L1 (EXTSUB)
      Application procedure (2).
INCLUDE  DFHSHRE (PLISHRE)
      Optional.  Use if shared library is to be used.
NAME     APROG (R)
      Optional.  Defines name of load module.
```

For more information about preparing PL/I programs, see the *OS PL/I Version 2 Programming Guide*, SC26-4307-02.

Sample JCL to install PL/I application programs

You can use the job control statements shown in Figure 16 on page 49 to process PL/I application programs. In the procedure name, “wx” depends on whether your programs are to be compiled by an LE-conforming compiler or otherwise, and whether it is a CICS application program or an EXCI batch program. For the names of the CICS-supplied procedures, see Table 7 on page 24.

```
//jobname    JOB  accounting info,name,MSGLEVEL=1
//           EXEC PROC=DFHwxTPL                                        1
//TRN.SYSIN  DD      *                                                2
*PROCESS    XOPTS(translator options...)PL/I compiler options...;     3
              .
             PL/I source statements                                   4
               .
/*
//LKED.SYSIN DD      *                                                5
               NAME    anyname(R)
/*
//
```

where `anyname` is your load module name

Figure 16. Sample job control statements to call the DFHwxTPL procedures

Notes for installing a PL/I program:

These notes apply to the DFHEITPL procedure only.

1 In the DFHEITPL procedure, the link-edit step includes module DFHPL1OI. This module is generated during the installation of PL/I and is normally placed either in the CICSTS13.CICS.SDFHLOAD library, the SYS1.PLIBASE library, or a user library. Find out which library contains this module, and either copy the module to the CICSTS13.CICS.SDFHLOAD library or concatenate the appropriate library in LKED.SYSLIB.

If you include the PL/I REPORT and COUNT execution time options, output goes to the CPLI transient data destination. There is an example of this transient data queue coded in the sample destination control table in the CICSTS13.CICS.SDFHSAMP library.

2 If you have no input for the translator, you can specify `DD DUMMY` instead of `DD *`. However, if you specify `DD DUMMY`, also code a suitable DCB operand. (The translator does not supply all the data control block information for the SYSIN data set.)

3 Translator and compiler options:

For information about the translator options you can include on the XOPTS statement, see the Specifying translator options in the *CICS Application Programming Guide* .

Ignore the message from the PL/I compiler: "`IEL0548I PARAMETER TO MAIN PROCEDURE NOT VARYING CHARACTER STRING`".

Warning messages may appear from the PL/I compiler stating that arguments and parameters do not match for calls to procedure DFHxxxx. These messages indicate that arguments specified in operands to CICS commands may not have the correct data type. Carefully check all fields mentioned in these messages, especially **receiver** fields.

4 If you include the CALL PLIDUMP statement in an application program, output goes to the CPLD transient data destination. The CICS supplied resource definition group, in the CSD, DFHDCTG, contains an entry for CPLD.

5 Linkage editor considerations:

You can ignore weak external references unresolved by the linkage editor.

If you are installing a program into either of the read-only DSAs, see “Preparing application programs to run in the RDSAs” on page 37 for more details.

If you are installing a program that is to be used from the LPA, add the RENT and REFR options to the LNKPARM parameter on the call to the DFHEITPL procedure. (See “Preparing applications to run in the link pack area” on page 36 for more information.)

To use the PL/I shared library facility, generate the module PLISHRE (see “Generating PL/I shared library support for CICS” on page 33) before you compile and link-edit your application programs. When you have re-link-edited the PLISHRE module into the CICSTS13.CICS.SDFHLOAD library, put the INCLUDE SYSLIB(PLISHRE) control statement immediately after the INCLUDE SYSLIB(DFHPL1OI) statement in the DFHEILIP member in the CICSTS13.CICS.SDFHPL1 library. Also, code a LKED.SYSLIB DD statement to concatenate the library that contains the PLISHRE module in front of the SYS1.PLIBASE library. For example:

```
//LKED.SYSLIB   DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//              DD DSN=SYS1.PLIBASE,DISP=SHR
```

Installing C application programs

Figure 17 on page 51 illustrates the flow of control in the cataloged procedures for C command-level programs.

Before you can install any C programs, have installed the C library and compiler and generated CICS support for C. (See “Installing CICS support for C/370” on page 32.)

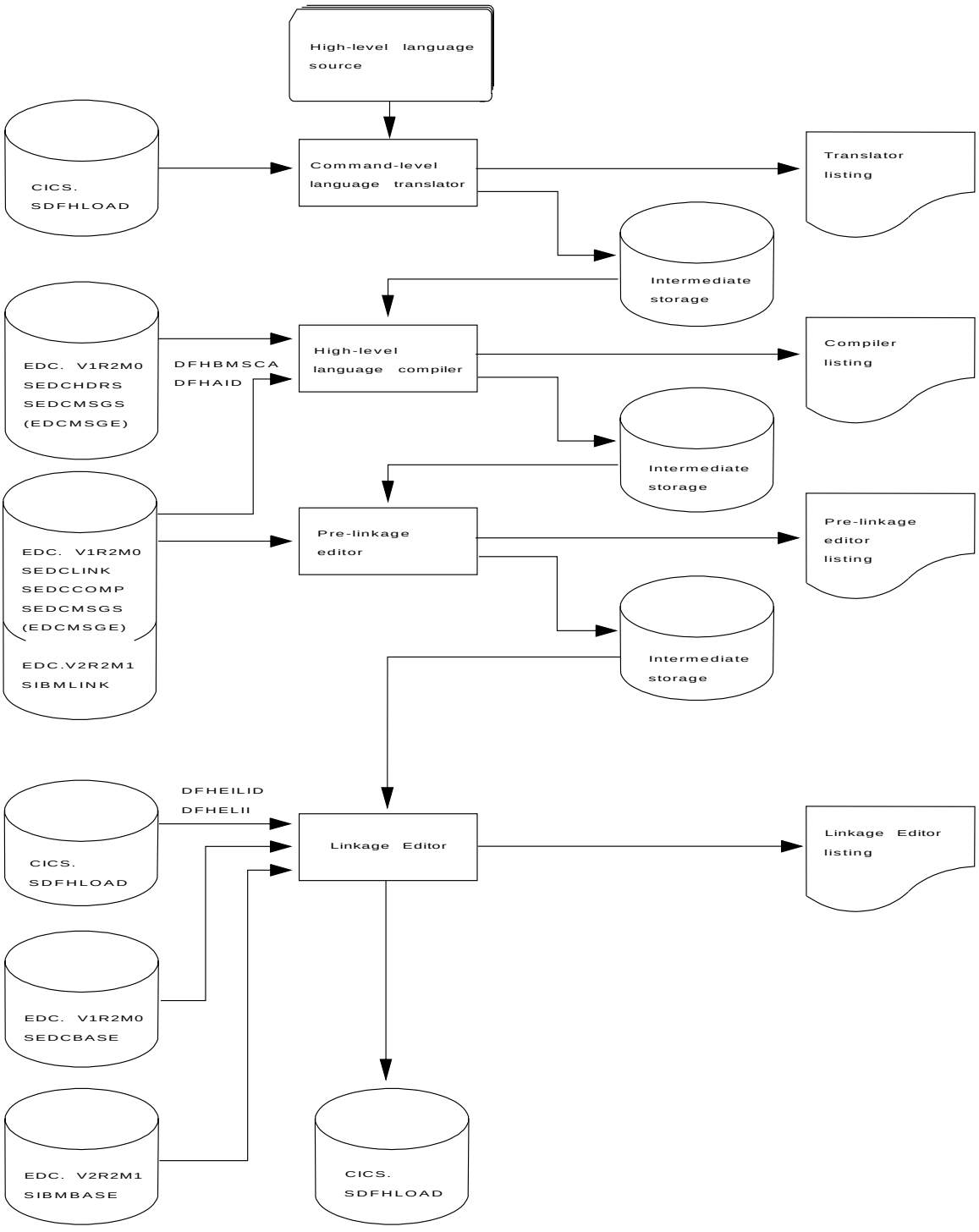


Figure 17. Installing C programs using the DFHxyTzL procedure

Sample JCL to install C application programs

You can use the job control statements shown in Figure 18 on page 52 to process C application programs. In the procedure name, `w` and `x` depend on whether your programs are to be compiled by an LE-conforming compiler or otherwise, and

whether it is a CICS application program or an EXCI batch program. For the names of the CICS-supplied procedures, see Table 7 on page 24.

```
//jobname    JOB  accounting info,name,MSGLEVEL=1
//           EXEC PROC=DFHwxTDL                                    1
//TRN.SYSIN  DD      *                                            2
#pragma XOPTS(Translator options . . .)                          3
             .
             C source statements
              .
/*
//LKED.SYSIN DD      *                                            4
               NAME    anyname(R)
/*
//
```

where `anyname` is your load module name

Figure 18. Sample JCL to call the DFHwxTDL procedures

Notes for installing a C program:

1 Compiler options:

You can code compiler options by using the parameter override (PARM.C) in the EXEC statement that invokes the procedure, or by a C statement at the start of the source statements. If you use a C statement, you need a parameter override of BATCH on the EXEC PROC=DFHEITyL statement.

You can compile your C/370 applications under Version 1 Release 2 of the C/370 compiler and run them under Version 1 Release 2 of the C/370 library.

2 If you have no input for the translator, you can specify `DD DUMMY` instead of `DD *`. However, if you specify `DD DUMMY`, also code a suitable DCB operand. (The translator does not supply all the data control block information for the SYSIN data set.)

3 Translator options: For information about the translator options you can include on the XOPTS statement, see the Specifying translator options*CICS Application Programming Guide* in the .

4 If you are installing a program into either of the read-only DSAs, see “Preparing application programs to run in the RDSAs” on page 37 for more details.

If you are installing a program that is to be used from the LPA, add the RENT and REFR options to the LNKPARM parameter on the call to the DFHEITyL procedure. (See “Preparing applications to run in the link pack area” on page 36 for more information.)

C language programs must be link-edited with AMODE(31), so the DFHwxTDL procedures specify AMODE(31) by default.

Using your own job streams

If you want to write your own JCL to translate, assemble (or compile), and link-edit your application programs, you can use the supplied cataloged procedures as a model. They are installed in the CICSTS13.CICS.SDFHPROC library.

The rest of this section summarizes the important points about the translator and each of the main categories of program. For simplicity, the following discussion states that you load programs into CICSTS13.CICS.SDFHLOAD or IMS.PGMLIB. In fact, you can use any libraries, but only when they are either included in the DFHRPL library concatenation in the CICS job stream, or included in the STEPLIB library concatenation in the batch job stream (for a stand-alone IMS batch program).

Note: The IMS libraries referred to in the job streams are identified by IMS.libnam (for example IMS.PGMLIB). If you use your own naming convention for IMS libraries, please rename the IMS libraries accordingly.

Translator requirements

The CICS translator requires a minimum of 256KB of virtual storage. You may need to use the translator options CICS and DLI.

If you have no input for the translator, you can specify `DD DUMMY` instead of `DD *`. However, if you specify `DD DUMMY`, also code a suitable DCB operand. (The translator does not supply all the data control block information for the SYSIN data set.)

Online programs that use EXEC CICS or EXEC DLI commands

1. Always use the translator option CICS. If the program issues EXEC DLI commands, use the translator option DLI.
2. The linkage editor input (defined by the SYSLIN DD statement) must include the correct interface module **before** the object deck. Therefore, place an INCLUDE statement for the interface module before the object deck. Also put ORDER statements before the INCLUDE statements, and an ENTRY statement after all the INCLUDE statements. The interface modules are:

   DFHEAI
   Assembler

   DFHELII
   C

   DFHECI
   COBOL

   DFHPL1OI
   PL/I

   DFHELII
   LE-conforming compilers

Note: Assembler and PL/I programs require additional modules, DFHEAI0 and DFHEPI respectively. In each case, inclusion of the interface module named above generates a reference to the additional module, and it is automatically included.

In the CICS-supplied procedures, the input to the linkage editor step (defined by the SYSLIN DD statement) concatenates a library member with the object deck. This member contains an INCLUDE statement for the required interface module. For example, the DFHEITVL procedure concatenates the library member DFHEILIC, which contains the following INCLUDE statement:

```
INCLUDE SYSLIB(DFHECI)
```

3. Place the load module output from the linkage editor (defined by the SYSLMOD DD statement) in CICSTS13.CICS.SDFHLOAD, or a user-defined application program library.

Figure 19 shows sample JCL and an inline procedure, based on the CICS-supplied procedure DFHEITVL, that can be used to install VS COBOL II application programs. The procedure does not include the COPYLINK step and concatenation of the library member DFHEILIC that contains the INCLUDE statement for the required interface module (as included in the DFHEITVL procedure). Instead, the JCL provides the following INCLUDE statement:

```
INCLUDE SYSLIB(DFHECI)
```

If this statement was not provided, the linkage editor would return an error message for unresolved external references, and the program output would be marked as not executable.

```
//COB2APPL JOB accounting info,name,MSGCLASS=A,MSGLEVEL=(1,1),
 //         CLASS=A
 //*
 //*********************************************************************
 //* THIS JOB CAN BE USED TO INSTALL A VS COBOL II PROGRAM.
 //*
 //* THE JOB ILLUSTRATES THE USE OF:
 //*
 //* 1) AN INLINE PROCEDURE BASED ON THE CICS=SUPPLIED PROCEDURE, DFHEITVL
 //*
 //* 2) INCLUDE AND ORDER STATEMENTS FOR INTERFACE MODULE DFHECI
 //*
 //*********************************************************************
 //*
 //MYEITVL PROC SUFFIX=1$,
 // INDEX='CICSTS13.CICS',
 // INDEX2='CICSTS13.CICS',
 //       OUTC=A,
 //       REG=5M,
 //       LNKPARM='XREF,RENT,REFR',
 //       WORK=SYSDA
 //*
 //*      THIS PROCEDURE CONTAINS 3 STEPS
 //*      1.   EXEC THE COBOL TRANSLATOR (USING SUFFIX 1$)
 //*      2.   EXEC THE VS COBOL II COMPILER
 //*      3.   LINKEDIT THE OUTPUT TO SDFHLOAD LIBRARY
 //*
```

Figure 19. Sample user-defined JCL to install a VS COBOL II program (Part 1 of 2)

```
//TRN    EXEC PGM=DFHECP&SUFFIX,
//       PARM='COBOL2',
//       REGION=&REG
//STEPLIB DD DSN=&INDEX..SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC
//SYSPUNCH DD DSN=&&SYSCIN,
//       DISP=(,PASS),UNIT=&WORK,
//       DCB=BLKSIZE=400,
//       SPACE=(400,(400,100))
//COB    EXEC PGM=IGYCRCTL,REGION=&REG,
//       PARM='NODYNAM,LIB,OBJECT,RENT,RES,APOST,MAP,XREF'
//STEPLIB DD DSN=PP.COB2.COB2COMP,DISP=SHR
//SYSLIB DD DSN=&INDEX..SDFHCOB,DISP=SHR
//       DD DSN=&INDEX..SDFHSAMP,DISP=SHR
//*      DD DSN=&INDEX2..SAMPLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//SYSLIN DD DSN=&&LOADSET,DISP=(MOD,PASS),
//       UNIT=&WORK,SPACE=(80,(250,100))
//SYSUT1 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT2 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT3 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT4 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT6 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT7 DD UNIT=&WORK,SPACE=(460,(350,100))
//*
//LKED   EXEC PGM=IEWL,REGION=&REG,
//       PARM='&LNKPARM',
//       COND=(5,LT,COB)
//SYSLIB DD DSN=&INDEX..SDFHLOAD,DISP=SHR
//       DD DSN=SYS2.COB2.COB2CICS,DISP=SHR
//       DD DSN=SYS2.COB2.COB2LIB,DISP=SHR
//* SYSLMOD DD DSN=&INDEX2..LOADLIB,DISP=SHR
//SYSLMOD DD DSN=&INDEX2..SDFHLOAD,DISP=SHR
//SYSUT1  DD UNIT=&WORK,DCB=BLKSIZE=1024,
//       SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSLIN DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//       DD DDNAME=SYSIN
// PEND
//*
//COB2TEST  EXEC MYEITVL,
//             INDEX='CICSTS13.CICS', QUALIFIER FOR CICS LIBRARIES
//             INDEX2='user.qualif',    QUALIFIER FOR USER LIBRARIES
//             OUTC='*',
//             REG='2048K',          SYSOUT CLASS
//             WORK='SYSDA',      UNIT FOR TEMPORARY DATA SETS
//             LNKPARM='XREF,LIST,RENT'
//TRN.SYSIN DD *
        ...
        VS COBOL II source statements
        ...
/*
//LKED.SYSIN DD *
  INCLUDE SYSLIB(DFHECI)
  NAME name(R)
/*
//
```

Figure 19. Sample user-defined JCL to install a VS COBOL II program (Part 2 of 2)

Online programs that use the CALL DLI interface

1. Specify the translator option CICS, but not the translator option DLI.

Note: For a program that does not use CICS commands and is only invoked by a running transaction (and never directly by CICS task initiation), no translator step is needed.

2. The interface module, DFHDLIAI, is automatically included by the linkage editor. If you use an INCLUDE statement in the linkage editor input, place it **after** the object deck.
3. Include the module DLIUIB.
4. Place the load module output from the linkage editor (defined by the SYSLMOD DD statement) in CICSTS13.CICS.SDFHLOAD, or a user-defined application program library.

Batch or BMP programs that use EXEC DLI commands

1. The translator option DLI is required. Do not specify the translator option CICS.
2. The INCLUDE statement for the interface module must **follow** the object deck in the input to the linkage editor (defined by the SYSLIN DD statement). The interface module, DFSLI000, which resides on IMS.RESLIB, is the same for all programming languages. If you include CICSTS13.CICS.SDFHLOAD in the input to the linkage editor (defined by the SYSLIB DD statement), concatenate it **after** IMS.RESLIB.
3. Place the load module output from the linkage editor (defined by the SYSLMOD DD statement) in IMS.PGMLIB, or a library concatenated in the STEPLIB DD statement of the batch job stream.

Batch or BMP programs that use DL/I CALL commands

If you want to prepare assembler, COBOL, or PL/I programs that use the DL/I CALL interface, do not use any of the CICS-supplied procedures. Programs that contain CALL ASMTDLI, CALL CBLTDLI, or CALL PLITDLI should be assembled or compiled, and link-edited, as IMS applications, and are not subject to any CICS requirements. See the relevant IMS manual for information about how to prepare application programs that use the DL/I CALL interface.

Defining programs, map sets, and partition sets to CICS

To be able to use a program that you have installed in one of the load libraries specified in your CICS startup JCL, the program, and any map sets and partition sets that it uses, must be defined to CICS. To do this, CICS uses the resource definitions MAPSET (for map sets), PARTITIONSET (for partition sets), and PROGRAM (for programs). You can create and install such resource definitions in any of the following ways:

- CICS can dynamically create, install, and catalog a definition for the program, map set, or partition set when it is first loaded, by using the autoinstall for programs function.
- You can create a specific resource definition for the program, map set, or partition set and install that resource definition in your CICS region.

  You can install resource definitions in either of the following ways:

  – At CICS initialization, by including the resource definition group in the group list specified on the GRPLIST system initialization parameter
  – While CICS is running, by the CEDA INSTALL command

For information about defining programs to CICS, see the *CICS Resource Definition Guide*.

Chapter 5. Defining DL/I support

This chapter describes what you do to enable a CICS region to work with remote DL/I. For information about adding system and resource definitions for use with DBCTL, see the *CICS IMS Database Control Guide*.

CICS can provide DL/I database support by using the IBM product Information Management System/Enterprise Systems Architecture (IMS/ESA) Database Manager Version 3 (5665-408) Release 1 or later.

Note: CICS/ESA 4.1 was the last release to support local DL/I.

You can use DL/I support with CICS through:

- Database control (DBCTL)
- CICS remote DL/I support, also known as **function shipping**

The IMS libraries referred to in the job streams are identified by IMS.libnam (for example IMS.PGMLIB). If you use your own naming convention for IMS libraries, please rename the IMS libraries accordingly.

Note: Not all releases of IMS can be used with the storage protection facilities available in CICS. There are restrictions on the use of storage protection when running DBCTL. Table 10 summarizes the IMS releases that can be used with storage protection.

Table 10. Summary of IMS releases that can be used with storage protection

Release of IMS	CICS with DBCTL
IMS/ESA 3.1	Storage protection not available
IMS/ESA 4.1	Storage protection
Later IMS/ESA releases	Storage protection

For more information about storage protection, see “Storage protection” on page 353.

PDIRs

A directory of program specification blocks (PDIR) is a list of program specification blocks (PSBs) that define, for DL/I, the use of databases by application programs.

Your CICS region needs a PDIR to access a database owned by a remote CICS region (remote DL/I support). Your CICS region does not need a PDIR to access a DL/I database owned by DBCTL. For information about accessing DL/I databases owned by DBCTL, see the *CICS IMS Database Control Guide*.

Adding remote DL/I support

Remote DL/I support is included in CICS Transaction Server for OS/390, and works with IMS 3.1, 4.1, and 5.1 (or later). Usually, you use remote DL/I support, with either MRO or ISC connections, to access databases owned by another CICS region. You can also use CICS remote DL/I support to access, via another CICS region connected to DBCTL, databases owned by DBCTL. CICS regions accessing

databases owned by DBCTL (that is, connected to DBCTL) must be running on the same MVS image as the DBCTL system. A simple overview is given in Figure 20 .

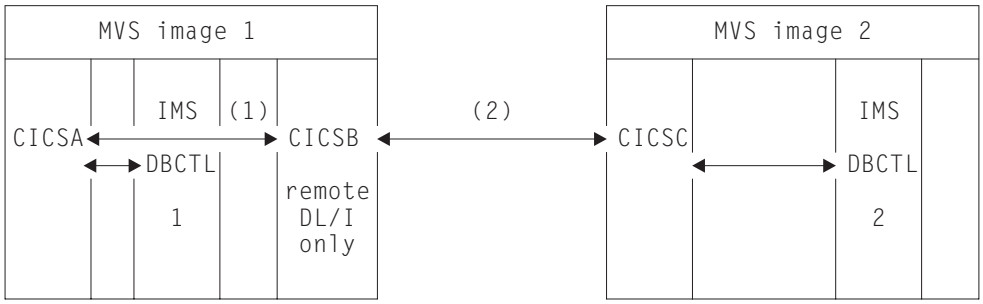


Figure 20. Using CICS remote DL/I support to access DBCTL databases

Notes:

1. CICSB uses remote DL/I to access, via CICSA, databases owned by DBCTL 1 in MVS image 1. This is only needed if CICSB is not connected to DBCTL 1.
2. CICSB uses remote DL/I to access, via CICSC, databases owned by DBCTL 2 in MVS image 2.
3. CICSA (connected to DBCTL 1) is in the same MVS image as DBCTL 1. CICSC (connected to DBCTL 2) is in the same MVS image as DBCTL 2.

For information about accessing DL/I databases owned by DBCTL, see the *CICS IMS Database Control Guide*.

To add support in CICS for remote database access, you must:

1. Code, assemble, and link-edit a program specification blocks directory (PDIR).
2. Code the PDIR CICS system initialization parameter for remote DL/I support.

Defining a PSB directory

You must code entries in a program specification block directory (PDIR), to indicate the identity of the remote CICS region, or regions, to which you want CICS to function ship DL/I requests. You do this by coding the SYSIDNT parameter in DFHDLPSB TYPE=ENTRY macros, which you assemble and link-edit to create a PDIR. You must also code the MXSSASZ parameter. You can, optionally, code the RMTNAME parameter to define the name by which the PSB is known in the remote CICS region. For information about creating PDIRs, see the *CICS Resource Definition Guide*.

Coding CICS system initialization parameters for remote DL/I support

The following is a summary of the DL/I parameters that you can, or must, code as CICS system initialization parameters:

```
PDIR={YES|xx}       SUFFIX OF PSB DIRECTORY
                                    (MANDATORY for REMOTE DL/I)
PSBCHK={NO|YES}     SECURITY CHECK OF REMOTE TERMINAL INITIATING A TRANSACTION
XPSB={YES|name|NO} PSB ENTRIES TO BE CHECKED BY RACF
```

For details of these (and other) system initialization parameters, see “Chapter 21. CICS system initialization parameters” on page 215.

Chapter 6. Defining DB2 support

Information about defining DB2 support is in the *CICS DB2 Guide*.

Chapter 7. 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 21. CICS system initialization parameters” on page 215.

   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 74.

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 OS/390 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 62.

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 via a local control unit.

Remote VTAM terminals
are attached to an SNA cluster controller, which is connected via 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 [2] 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 302 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

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.

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.

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.

TCAM terminals

CICS supports the DCB interface of ACF/TCAM (also known as the GET/PUT interface) in an SNA or non-SNA environment. This section describes the DFHTCT macros you must code to define the CICS terminals connected to this interface.

With the CICS support of the TCAM DCB interface, each TCAM communication line has associated with it two “sequential” queues: the input process queue and the output process queue. CICS routes messages for terminals connected using the TCAM DCB interface to the queue named in the DEST option of the SEND and CONVERSE commands.

CICS assumes that there is a user-written message control program (MCP) that processes messages on the TCAM queues. The TCAM MCP is responsible for polling and addressing terminals, translating code, and line control. Therefore, some DFHTCT operands that are associated with such activities are irrelevant in the TCAM DCB interface environment.

For programming information about the CICS/TCAM interface, see the *CICS Customization Guide*.

You code one DFHTCT TYPE=SDSCI macro for each input queue, and one for each output queue. The macros generate DCBs, corresponding to TPROCESS blocks. CICS treats a queue like a communication line. Each queue is described by a DFHTCT TYPE=LINE macro; this generates one TCT line entry (TCTLE) for each queue.

Note: TCAM DCBs are stored below the 16MB line.

Each TCAM terminal needs a DFHTCT TYPE=TERMINAL macro; this generates one TCT terminal entry (TCTTE) for each terminal. To avoid duplicating the TCTTEs for both the input queue and the output queue, describe all the terminals immediately after the DFHTCT TYPE=LINE macro for the output queue. Although attached to the output TCTLE, these TCTTEs are used for both input and output

processing. You must also generate one dummy TCTTE for the input TCTLE; this need have only a TRMIDNT operand giving a dummy terminal identification and a LASTTRM operand.

Each input record from TCAM must contain the source terminal identification. CICS uses this identification as a search argument to find the corresponding TCTTE (by comparing against the NETNAME value for each TCTTE).

Note: The usual way to ensure that the input records contain the source terminal identification is to specify OPTCD=W in the DFHTCT TYPE=SDSCI macro. If you omit this specification, you are responsible for ensuring that the record contains a suitable source terminal identification.

By using the POOL feature (POOL=YES on the DFHTCT TYPE=LINE macro), you can establish a pool of common TCTTEs on the output TCTLE that do not contain terminal identifiers. As required, terminal identifiers are assigned to the TCTTEs or removed from association with the TCTTEs. For programming information about the TCTTEs, see the *CICS Customization Guide*.

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 CICSTS13.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 CICSTS13.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=volid
//DISKOT1  DD DSN=SIMULATD.TERMINAL.OUT,
//         UNIT=3380,DISP=OLD,VOL=SER=volid
```

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 21. CICS system initialization parameters” on page 215 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 21 on page 67, and add the required DD statement to the CICS startup JCL.

```
*
DFHTCT TYPE=INITIAL,                    X
               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 21. 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 21

```
//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 11 at end-of-file on the sequential device:

Table 11. Warning messages if a sequential terminal is not closed

Message	***Destination***
DFHTC2507 *date time applid* Input event rejected return code *zz* {on line w/term|at term}*termid* {, trans}*tranid*{, rel line=} *rr,time*	CSMT
DFHTC2500 *date time applid* {Line|CU|Terminal} out of service {Term|W/Term} *termid*	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 11 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, and change the status from RECEIVE to TRANSCEIVE **after** CICS initialization by using the CEMT master terminal transaction (input through the console or a master terminal):

  ```
  CEMT SET TERMINAL(termid) TTI
  ```
- 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.

Console devices

You can operate CICS from a **console device** [3] .

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, CEBT.

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 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=YES|AUTO 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*.

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 CONSOL*nn* 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 70.

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 continues to support the CONSOLE(*number*) attribute, but this is not recommended, because the number is dynamically allocated by MVS during IPL, and in a sysplex configuration these numbers can vary from IPL to IPL. (The identification numbers are determined by the sequence in which they are encountered in the several CONSOLnn members for the MVS images. As a consequence, the results of using CONSOLE may be unpredictable.) 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 22.

```
//DEFTERM  JOB (accounting information),MSGCLASS=A,
//         MSGLEVEL=(1,1),CLASS=A,NOTIFY=userid
//CONSDEF  EXEC PGM=DFHCSDUP
//STEPLIB  DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=CICSTS13.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 22. 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 22.

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 22 on page 70.

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.

DFHTERMC
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 session 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 session 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 session 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 OS/390 Release 3 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*.

Chapter 8. Defining sequence numbering resources

This chapter describes the steps required to support a named counter server. It covers the following topics:

Overview

CICS provides a facility for generating unique sequence numbers for use by applications in a Parallel Sysplex environment (for example, to allocate a unique number for orders or invoices). This facility is provided by a named counter server, which maintains each sequence of numbers as a named counter. Each time a sequence number is assigned, the corresponding named counter is incremented automatically so that the next request gets the next number in sequence.

The named counter server is modeled on the other coupling facility servers used by CICS, and has many features in common with the coupling facility data table server.

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.

For information on how to create a list structure for use as a named counter pool, see "Defining a list structure" on page 90.

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.

For information about creating a loadable options table, see “Defining a named counter options table” on page 87.

The named counter application programming interface

You access the named counter through a callable interface, which can be used in CICS applications running in CICS key or user key, or used in batch jobs. The interface does not depend on CICS services, therefore it can also be used in applications running under any release of CICS.

The named counter interface does not use CICS command-level API, therefore the system initialization parameter CMDPROT=YES has no effect. If the interface is called from a CICS application program that is excuting in user key, it switches to CICS key while processing the request but CICS does not attempt to verify that the program has write access to the output parameter fields.

The first request by a CICS region that addresses a particular pool automatically establishes a connection to the server for that pool. This connection is associated with the current MVS TCB (which for CICS is the quasi-reentrant (QR) TCB) and normally lasts until the TCB terminates at end of job. An application region can have only one connection at a time to each named counter pool, and the connection can be used only from the TCB under which the connection was established. A connection can only be created under another TCB by first terminating the existing connection using the NC_FINISH function, and then creating a new connection from another TCB.

Note: The named counter server interface uses MVS name/token services internally. A consequence of this is that jobs using the named counter interface cannot use MVS checkpoint/restart services (as described in APAR OW06685).

Application programming considerations

To use the named counter callable interface:

1. Ensure your application programs include the appropriate copybook that defines the parameter list definition for the application programming language. The copybook defines symbolic constants for the function codes and return codes, and also defines the callable entry point for high level languages. The copybook name is of the form DFHNC*xxx* where *xxx* indicates the programming language, as follows:

   ASM or **EQU** for Assembler
   C for C/C++
   COB for COBOL
   PLI for PL/I

2. Ensure the application program is link-edited with the callable interface linkage routine, DFHNCTR.
3. Ensure the named counter server interface module, DFHNCIF, and the options table, DFHNCOPT, are available to the CICS region. That is, these objects must be in a STEPLIB library, in a linklist library, or in the LPA. To support CICS

application programs that run in user key, DFHNCIF must be loaded from an APF-authorized library. The default option table and the named counter server interface module are supplied in CICSTS13.CICS.SDFHLINK.

CICS provides copybooks for all the supported languages:

Assembler

The standard assembler named counter interface definitions are provided in copybook DFHNCASM. Include these in your application programs using COPY DFHNCASM within a constant CSECT area. The symbolic values are defined as static fullword constants, in fhe form NC_*name* DC F'*nnn*'. For example:

```
NC_BROWSE_NEXT DC F'7'
```

An alternative set of definitions is provided as symbolic equated values in copy book DFHNCEQU. These symbols are all of the form NC_EQU_*name* to avoid conflict with the static constants. Note that when these equated values are used for function codes or return code comparisons, they should be used as address constant values, so that for example the function code NC_ASSIGN can be replaced by a reference to =A(NC_EQU_ASSIGN).

The syntax of the assembler version of the call to the named counter interface is as follows:

```
CALL DFHNCTR,(function,return_code,pool_selector,counter_name,        X
            value_length,current_value,minimum_value,maximum_value,   X
            counter_options,update_value,compare_min,compare_max),VL
```

The CALL macro must specify the VL option to set the end of list indication, as shown in the following example:

```
CALL DFHNCTR,(NC_ASSIGN,RC,POOL,NAME,CTRLEN,CTR),VL
```

C/C++

The named counter interface definitions for C/C++ are provided in header file DFHNCC. The symbolic constant names are in upper case. The function name is `dfhnctr`, in lower case.

COBOL

The named counter interface definitions for COBOL are provided in copybook DFHNCCOB.

COBOL does not allow underscores within names, therefore the symbolic names provided in copy book DFHNCCOB use a hyphen instead of an underscore (for example NC-ASSIGN and NC-COUNTER-AT-LIMIT).

Note that the RETURN-CODE special register is set by each call, which affects the program overall return code if it is not explicitly set again before the program terminates.

PL/I

The named counter interface definitions for PL/I are provided in include file DFHNCPLI.

Syntax

The syntax of the named counter call is as follows:

```
CALL DFHNCTR(function,return_code,pool_selector,counter_name,
             value_length,current_value,minimum_value,maximum_value,
             counter_options,update_value,compare_min,compare_max);
```

Figure 23. DFHNCTR call syntax-PL/I illustration

Notes:

1. All functions that refer to a named counter require at least the first four parameters, but the remaining parameters are optional, and trailing unused parameters can be omitted.

   If you do not want to use an imbedded optional parameter, either specify the default value or ensure that the parameter list contains a null address for the omitted parameter. For an example of a call that omits an optional parameter, see “Example of DFHNCTR calls with null parameters” on page 84.

2. The NC_FINISH function requires the first three parameters only.

function
specifies the function to be performed, as a 32-bit integer, using one of the following symbolic constants.

NC_CREATE Create a new named counter, using the initial value, range limits, and default options specified on the *current_value, minimum_value, maximum_value, update_value* and *counter_options* parameters.

If you omit an optional value parameter, the new named counter is created using the default for the omitted value. For example, if you omit all the optional parameters, the counter is created with an initial value of 0, a minimum value of 0, and a maximum value of high values (the double word field is filled with X'FF').

NC_ASSIGN Assign the current value of the named counter, then increment it ready for the next request. When the number assigned equals the maximum number specified for the counter, it is incremented finally to a value 1 greater than the maximum. This ensures that any subsequent NC_ASSIGN requests for the named counter fail (with NC_COUNTER_AT_LIMIT) until the counter is reset using the NC_REWIND function, or automatically rewound by the NC_WRAP counter option (see the *counter_options* parameter).

This operation can include a conditional test on the current value of the named counter, using the *compare_min* and *compare_max* parameters.

The server returns the minimum and maximum values if you specify these fields on the call parameter list and the request is successful.

You can use the *counter_options* parameter on the NC_ASSIGN request to override the counter options set by the NC_CREATE request.

You can use the *update_value* parameter to specify the increment to be used on this request only for the named

counter (the default increment is 1). This enables you to obtain a range of numbers on a single request. For more information, see the description of the *update_value* parameter.

Note that the named counter is incremented by *update_value* after the current value is assigned. For example:

If the current value is 109
and *update_value* specifies 25

the named counter server returns 109 and sets the current value to 134 ready for the next NC_ASSIGN request, effectively assigning numbers in the range 109 through 133. The increment can be any value between zero and the limit is determined by the minimum and maximum values set for the named counter. Thus the increment limit is ((*maximum_value* plus 1) minus *minimum value*). An increment of zero causes NC_ASSIGN to operate the same as NC_INQUIRE, except for any comparison options.

When the increment is greater than 1, and the named counter is near the maximum limit, the server may not be able to increment the current number by the whole value of the specified increment. This situation occurs when incrementing the counter would exceed the maximum value plus 1. You control what action the named counter server takes in this situation through the *counter_options* NC_NOREDUCE | NC_REDUCE, and NC_NOWRAP | NC_WRAP. See *counter_options* parameter for information about the operation of these options.

NC_BROWSE_FIRST
Return the details of the first named counter with a name greater than or equal to the specified name, and update the *counter_name* field accordingly.

NC_BROWSE_NEXT
Return the details of the next named counter with a name greater than the specified name, and update the *counter_name* field accordingly.

NC_DELETE Delete the specified named counter. The server returns the *current_value, minimum_value, maximum_value*, and *counter_options* if you specify these fields on the parameter list and the request is successful.

NC_FINISH Terminate the connection between the current MVS task (TCB) and the named counter server for the specified pool. If a further request is made to the same pool, a new connection is established.

This function does not apply to a specific counter, therefore the only parameters required are the function, the return code and the pool name.

Use this function only when there is a special reason to terminate the connection (for example, to allow the server to be shut down).

NC_INQUIRE Return the details (*current_value, minimum_value, maximum_value* and *counter_options*) of a named counter without modifying it. The current value is the value to be returned on the next NC_ASSIGN call. If the maximum value of the named counter has already been assigned, the server returns a current value one greater than the maximum value.

NC_REWIND Reset the named counter to its minimum value. This function is valid only when the last number permitted by the maximum value has been assigned, leaving the counter in the NC_COUNTER_AT_LIMIT condition. If an NC_ASSIGN call causes the server to assign the last number for the named counter, use the NC_REWIND function to reset the counter.

This operation can include a conditional test on the current value of the named counter, using the compare_min and compare_max parameters.

The server returns the new current value, minimum value, and maximum value if you specify these fields on the parameter list and the request is successful.

If any option parameter or *update_value* parameter was specified on an NC_ASSIGN request which failed because the named counter was at its limit, the same parameter values must also be specified on the NC_REWIND request, so that it can check whether the original NC_ASSIGN would still fail. The NC_REWIND request is suppressed with return code 102 (NC_COUNTER_NOT_AT_LIMIT) whenever the corresponding NC_ASSIGN request would succeed.

If the NC_WRAP option is in effect, or the *update_value* parameter is zero, NC_REWIND is suppressed because NC_ASSIGN always succeeds with these conditions. See the *counter_options* parameter for information about the NC_WRAP option.

NC_UPDATE Set the named counter to a new value. This operation can include a conditional test on the current value of the named counter, using the *compare_min* and *compare_max* parameters.

You specify the new value on the *update_value* parameter. If you don't specify a new value, the named counter remains unchanged.

You can specify a valid *counter_options* override parameter (or a null address) with this function, but counter options have no effect. Specify either a null address or NC_NONE as the *counter_options* parameter.

return_code
specifies a 32-bit integer field to receive the return code. The same information is also returned in register 15, which for COBOL callers is stored in the RETURN-CODE special register.

Each return code has a corresponding symbolic constant. See "Return codes" on page 85 for details of these.

pool_selector
specifies an 8-character pool selection parameter that you use to identify the pool in which the named counter resides.

This parameter is optional. If you omit the pool selection parameter, a string of 8 blank X'40') characters is assumed.

The acceptable characters for *pool_selector* are A through Z, 0 through 9, $ @ # and _ (underscore) but the first character cannot be numeric or an underscore. The parameter should be padded to fill 8 characters with trailing spaces as necessary. The parameter can be all spaces to use the default pool for the current region, provided this is mapped by the options table to a valid non-blank named counter name).

Depending on the named counter options table in use, you can use the pool selector parameter either as an actual pool name, or as a logical pool name that is mapped to a real pool name through the options table. The default options table assumes:

- That any pool selection parameter beginning with DFHNC (matching the table entry with POOLSEL=DFHNC*) is an actual pool name
- That any other pool selection parameter (including all blanks) maps to the default pool name.

Note: The default pool name for the call interface is DFHNC001. The default pool name for the EXEC CICS API is defined by the NCPLDFT system initialization parameter.

See “Defining a named counter options table” on page 87 for information about the pool selection parameter in the DFHNCOPT options table.

counter_name
specifies a 16-byte field containing the name of the named counter, padded if necessary with trailing spaces.

The acceptable characters for the name are A through Z, 0 through 9, $ @ # and _ (underscore) but the first character cannot be numeric or an underscore. The parameter should be padded to fill 16 characters with trailing spaces as necessary.

You are recommended to use names that have a common prefix of up to 8 bytes to avoid conflicts with other applications. Any named counters used internally by CICS have names starting with DFH.

For the NC_BROWSE_FIRST and NC_BROWSE_NEXT functions, the actual name is returned in this field, which must be a variable for this purpose. For all other functions, this can be a constant.

value_length
specifies a 32-bit integer field containing the length of each named counter value field in bytes, in the range 1 to 8. If no value parameters are used on the call, you can either omit all the trailing unused parameters completely, including the value length, or specify *value_length* as 0.

When input values are shorter than 8 bytes, they are extended with high-order zero bytes to the full 8 bytes used internally. When output values are returned in a short value field, the specified number of low-order bytes are returned, ignoring any higher-order bytes.

current_value
specifies a variable to be used for:

- Setting the initial sequence number for the named counter

- Receiving the current sequence number from the named counter.

For the NC_CREATE function this parameter is an input (sender) field and can defined as a constant. The default value is low values (binary zeroes). The value can either be within the range specified by the following minimum and maximum values, or it can be one greater than the maximum value, in which case the counter has to be reset using the NC_REWIND function before it is used.

For all other counter functions, this parameter is an output (receiver) field and must be defined as a variable.

minimum_value
specifies a variable to be used for:

- Setting the minimum value for the named counter
- Receiving from the named counter the specified minimum value.

For the NC_CREATE function this parameter is an input (sender) field and can defined as a constant.

For all other functions, this parameter is an output (receiver) field and must be defined as a variable.

maximum_value
specifies a variable to be used for:

- Setting the maximum value for the named counter
- Receiving from the named counter the specified maximum value.

For the NC_CREATE function this parameter is an input (sender) field and can defined as a constant. If you specify a non-zero *value_length* parameter but omit *maximum_value*, it defaults to high values for the specified length, otherwise it is eight bytes of high values. However, if the minimum value is all low values and the maximum value is eight bytes of high values, the maximum value is reduced to allow some reserved values to be available to the server for internal use.

For all other functions, this parameter is an output (receiver) field and must be defined as a variable.

counter_options

specifies an optional fullword field to indicate named counter options that control wrapping and increment reducing. The valid options are represented by the symbolic values NC_WRAP|NC_NOWRAP and NC_REDUCE|NC_NOREDUCE. The default options are NC_NOWRAP and NC_NOREDUCE.

NC_NOWRAP The server does not automatically rewind the named counter back to the minimum value in response to an NC_ASSIGN request that fails with the NC_COUNTER_AT_LIMIT condition. With NC_NOWRAP in force, and the named counter in the NC_COUNTER_AT_LIMIT condition, the NC_ASSIGN function is inoperative until the counter is reset by an NC_REWIND request (or the counter option reset to NC_WRAP).

NC_WRAP The server automatically performs an NC_REWIND in response an NC_ASSIGN request for a counter that is in the NC_COUNTER_AT_LIMIT condition. The server sets the

current value of the named counter equal to the minimum value, returns the new current value to the caller, and increments the named counter.

NC_NOREDUCE
If the range of numbers remaining to be assigned (the difference between the current value and the maximum value plus 1) is less than the increment specified on the *update_value* parameter, the assign fails (unless NC_WRAP is in force). NC_NOREDUCE, with NC_NOWRAP, means the NC_ASSIGN request fails with the NC_COUNTER_AT_LIMIT condition.

For example, if a request specifies an update value of 15 when the current number is 199 990 and the counter maximum number is defined as 199 999, the NC_REQUEST fails because the increment would cause the current number to exceed 200 000.

NC_REDUCE If the range of numbers remaining to be assigned (the difference between the current value and the maximum value plus 1) is less than the increment specified on the *update_value* parameter, the increment is reduced and the assign succeeds. In this case, the NC_ASSIGN request has been assigned a range of numbers less than that specified by the *update_value*, and the named counter is left in the NC_COUNTER_AT_LIMIT condition. Subsequent NC_ASSIGN requests will fail until the named counter is reset with an NC_REWIND request.

The options specified on NC_CREATE are stored with the named counter and are used as the defaults for other named counter functions. You can override the options on NC_ASSIGN, NC_REWIND or NC_UPDATE requests. If you don't want to specify *counter_options* on a DFHNCTR call, specify the symbolic constant NC_NONE (equal to zero) as the input parameter (or specify a null address).

For the NC_CREATE, NC_ASSIGN, NC_REWIND, and NC_UPDATE functions, this parameter is an input field.

For the NC_DELETE, NC_INQUIRE, and NC_BROWSE functions, this parameter is an output field, which returns the options specified on NC_CREATE.

update_value
specifies the value to be used to update the counter. For NC_ASSIGN, this is the increment to be added to the current counter value (after the current number is assigned). See the NC_ASSIGN option on the *function* parameter for information on how specifying an increment other than 1 can affect an assign operation.

For NC_UPDATE, this is the new current value for the named counter.

compare_min
specifies a value to be compared with the named counter's current value. If you specify a value, this parameter makes the NC_ASSIGN, NC_REWIND or NC_UPDATE operation conditional on the current value of the named counter being greater than or equal to the specified value. If the comparison is not satisfied, the operation is rejected with a counter-out-of-range return code (RC 103).

If you omit this parameter by specifying a null address, the server does not perform the comparison.

compare_max
specifies a value to be compared with the named counter's current value. If you specify a value, this parameter makes the NC_ASSIGN, NC_REWIND or NC_UPDATE operation conditional on the current value of the named counter being less than or equal to the specified value. If the comparison is not satisfied, the operation is rejected with a counter-out-of-range return code (RC 103).

If you specifying high values (X'FF') for this parameter, the server does not perform the comparison. You must specify (X'FF') in all the bytes specified by the *value_length* parameter.

If the compare_max value is less than the compare_min value, the valid range is assumed to wrap round, in which case the current value is considered to be in range if it satisfies either comparison, otherwise both comparisons must be satisfied.

Example of DFHNCTR calls with null parameters

If you omit an optional parameter on a DFHNCTR call, ensure that the parameter list is built with a null address for the missing parameter. The example that follows illustrates how to issue, from a COBOL program, an NC_CREATE request with some parameters set to null addresses.

DFHNCTR call with null addresses for omitted parameters: In this example, the parameters used on the call are defined in the WORKING-STORAGE SECTION, as follows:

Call parameter	COBOL variable	Field definition
function	01 FUNCTION	PIC S9(8) COMP VALUE +1.
return_code	01 NC-RETURN-CODE.	PIC S9(8) COMP VALUE +0.
pool_selector	01 NC-POOL-SELECTOR	PIC X(8).
counter_name	01 NC-COUNTER-NAME	PIC X(16).
value_length	01 NC_VALUE-LENGTH	PIC S9(8) COMP VALUE +4.
current_value	01 NC-CURRENT-VALUE	PIC S9(8) VALUE +0.
minimum_value	01 NC-MIN-VALUE	PIC S9(8) VALUE +0.
maximum_value	01 NC-MAX-VALUE	PIC S9(8) VALUE -1.
counter_options	01 NC-OPTIONS	PIC S9(8) COMP VALUE +0.
update_value	01 NC-UPDATE-VALUE	PIC S9(8) VALUE +1.
compare_min	01 NC-COMP-MIN	PIC S9(8) VALUE +0.
compare_max	01 NC-COMP-MAX	PIC S9(8) VALUE +0.

The variable used for the null address is defined in the LINKAGE SECTION, as follows:

```
LINKAGE  SECTION.
  01   NULL-PTR            USAGE  IS  POINTER.
```

Using the data names specified in the WORKING-STORAGE SECTION as described above, and the NULL-PTR name as described in the LINKAGE SECTION, the following illustrates a call to a named counter server where *value_length, current_value, minimum_value* and *counter_options* are the only optional parameters specified. The others are allowed to default, or, in the case of trailing optional parameters, omitted altogether.

```
 NAMED-COUNTER SECTION.
*
     SET ADDRESS OF NULL-PTR TO NULLS.
*
     MOVE   1     TO  FUNCTION.
     MOVE 100     TO  NC-MIN-VALUE  NC-CURRENT-VALUE.
     MOVE NC-WRAP TO  NC-OPTIONS.
     MOVE "DFHNC001" TO NC-POOL-SELECTOR.
     MOVE "CUSTOMER_NUMBER" TO NC-COUNTER-NAME.
     CALL 'DFHNCTR' USING  FUNCTION NC-RETURN-CODE NC-POOL-SELECTOR
                    NC-COUNTER-NAME  NC-VALUE-LENGTH  NC-CURRENT-VALUE
                    NC-MIN-VALUE  NULL-PTR  NC-OPTIONS.
```

Return codes

The return codes are divided into ranges (100, 200, 300, and 400) according to their severity. Each range of non-zero return codes begins with a dummy return code that describes the return code category, to make it easy to check for values in each range using a symbolic name.

In the list that follows, the numeric return code is followed by its symbolic name.

0 (NC_OK)
The request completed normally.

100 (NC_COND)
Return codes in this range indicate that a conditional function did not succeed because the condition was not satisfied:

101 (NC_COUNTER_AT_LIMIT)
An NC_ASSIGN function is rejected because the previous request for this named counter obtained the maximum value and the counter is now at its limit. New counter values cannot be assigned until an NC_REWIND function call is issued to reset the counter.

102 (NC_COUNTER_NOT_AT_LIMIT)
An NC_REWIND FUNCTION is rejected because the named counter is not at its limit value. This is most likely to occur when another task has lready succeeded in resetting the counter with an NC_REWIND.

103 (NC_COUNTER_OUT_OF_RANGE)
The current value of the named counter is not within the range specified on the *compare_min* and *compare_max* parameters.

200 (NC_EXCEPTION)
Return codes in this range indicate an exception condition that an application program should be able to handle:

201 (NC_COUNTER_NOT_FOUND)
The named counter cannot be found.

202 (NC_DUPLICATE_COUNTER_NAME)
An NC_CREATE function is rejected because a named counter with the specified name already exists.

203 (NC_SERVER_NOT_CONNECTED)
An NC_FINISH function is rejected because no active connection exists for the selected pool.

300 (NC_ENVIRONMENT_ERROR)
Return codes in this range indicate an environment error. These are serious errors, normally caused by some external factor, which a program may not to be able to handle.

301 (NC_UNKNOWN_ERROR)
The server has reported an error code that is not understood by the interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

302 (NC_NO_SPACE_IN_POOL)
A new counter cannot be created because there is insufficient space in the named counter pool.

303 (NC_CF_ACCESS_ERROR)
An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information can be found in message DFHNC0441 in the CICS job log.

304 (NC_NO_SERVER_SELECTED)
The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 (NC_SERVER_NOT_AVAILABLE)
The interface is unable to establish a connection to the server for the appropriate named counter pool. Further information can be found in an AXM services message in the CICS job log.

306 (NC_SERVER_REQUEST_FAILED)
An abend occurred during server processing of a request. Further information can be found in a message in the CICS job log and the server job log.

307 (NC_NAME_TOKEN_ERROR)
An IEANT*xx* name/token service call within the named counter interface module gave an unexpected return code.

308 (NC_OPTION_TABLE_NOT_FOUND)
The DFHNCOPT options table module, required for resolving a pool name, could not be loaded.

309 (NC_OPTION_TABLE_INVALID)
During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 (NC_USER_EXIT_NOT_FOUND)
An options table entry matching the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

400 (NC_PARAMETER_ERROR)
Return codes in this range indicate a parameter error, generally the result of a coding error in the calling program.

401 (NC_INVALID_PARAMETER_LIST)
The parameter list is invalid for one of the following reasons:

- Too few parameters are specified (less than four, or less than three for the NC_FINISH function)
- Too many parameters are given (more than eight)
- A parameter address is zero
- The end-of-list marker is missing.

402 (NC_INVALID_FUNCTION)
The function code parameter is not in the supported range.

403 (NC_INVALID_POOL_NAME)
The pool selection parameter contains characters that are not allowed, or embedded spaces.

404 (NC_INVALID_COUNTER_NAME)
The *counter_name* parameter contains characters that are not allowed, or embedded spaces.

405 (NC_INVALID_VALUE_LENGTH)
The value length parameter is not in the range 0 to 8.

406 (NC_INVALID_COUNTER_VALUE)
The specified counter value or increment value is inconsistent with the minimum and maximum limits for the counter.

The counter value specified on the *current_value* parameter for the NC_CREATE function, or the *update_value* for the NC_UPDATE function, cannot be less than the specified minimum value, and cannot be more than (maximum value + 1).

The increment value specified in the *update_value* parameter for the NC_ASSIGN or NC_REWIND function cannot be greater than the total range of the counter ((maximum value − minimum value) + 1).

407 (NC_INVALID_COUNTER_LIMIT)
The maximum value is less than the minimum value.

408 (NC_INVALID_OPTIONS)
The value of the *counter_options* parameter is invalid. It is either a value that does not correspond to any defined option, or it is a value that represents some mutually exclusive options.

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. 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 Table 12.

Table 12. The DFHNCO macro. For generating a DFHNCOPT options table

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

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, with a standard save area and parameter list pointing to two fields, in the following order:

- The 8-byte actual pool name result field
- The 8-byte pool selection parameter.

The end-of-list bit is set in the second parameter address.

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 CICSTS13.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 CICSTS13.CICS.SDFHSAMP.

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 24.

```
STRUCTURE  NAME(DFHNCLS_PRODNC1)
   SIZE(512)
   INITSIZE(256)
   PREFLIST(FACIL01,FACIL02)
```

Figure 24. Example of statements defining 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=policyname,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.

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 approximately one thousand named counters, and the next higher size of 512KB can hold some tens of thousands.

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.

Part 2. Defining data sets

Having defined your resources and installed your application programs (see “Part 1. Installing resource definitions” on page 1), you must now set up data sets and data definition statements. This part of the book consists of one chapter for each of the CICS facilities. Each chapter describes the facility, its function and usage, and the data sets needed to implement it on a running CICS region. If the data sets need preformatting, jobs that you can use for this purpose are shown.

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

Chapter 9. Preparing to set 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*.

Overview of setting up CICS data sets

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

- Set up and catalog the data sets and libraries that are used by the CICS programs 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
  - DL/I databases you are accessing through CICS remote DL/I support or DBCTL

  For more information about user file definitions, see “Chapter 18. Defining user files” on page 189.

Table 13 on page 94 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, CICSTS13.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 *MVS Sysplex Application Migration* manual, GC28-1211. [4] For example, if CICSHTH1 is the APPLID, the data set name for the CSD would be:

```
DFHCSD   DD  DSN=CICSTS13.CICS.CICSHTH1.DFHCSD,DISP=SHR
```

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 “Data set considerations when running CICS with XRF” on page 98.

Table 13. Summary of CICS data sets

Data set	DDNAME used by CICS	Block or control interval size (bytes)	Record format	Data set organization	Other comments
AUXILIARY TRACE (See page 171)	DFHAUXT DFHBUXT	4096	F	Sequential	3 See page 97 for information about GTF.
BTS LOCAL REQUEST QUEUE (See the *CICS Business Transaction Services* manual)	DFHLRQ	1024 and 2560	VB	VSAM KSDS	Required *even if you do not use BTS facilities*. BTS is described in the *CICS Business Transaction Services* manual.
CAVM CONTROL (See page 181)	DFHXRCTL	4096 minimum	1	VSAM ESDS	Required if running CICS with XRF.
CAVM MESSAGE (See page 181)	DFHXRMSG	4096 minimum	1	VSAM ESDS	Required if running CICS with XRF.
CATALOGS (See page 159)	DFHGCD DFHLCD	8192 & 2048	VB	VSAM KSDS	Both data sets must be initialized before use (2).
CDBM group command (See page 207)	DFHDBFK	8192	VB	VSAM KSDS	Required *only if you intend to use this function*.
CSD (See page 135)	DFHCSD	8192	VB	VSAM KSDS	—
DUMP (See page 175)	DFHDMPA DFHDMPB	32 760 (tape) or 1 track (DASD)	V	Sequential	For CICS transaction dumps only; see page 97 for information about system dumps.

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.

Table 13. Summary of CICS data sets (continued)

Data set	DDNAME used by CICS	Block or control interval size (bytes)	Record format	Data set organization	Other comments
MESSAGES (See page 211)	DFHCMACD	—	V	VSAM KSDS	Can be created and loaded by the DFHCMACI job.
TEMPORARY STORAGE (See page 107)	DFHTEMP	See page 108	1	VSAM ESDS	—
TRANSIENT DATA EXTRA PARTITION (See page 115) 4	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 INTRA-PARTITION (See page 118)	DFHINTRA	See page 116.	1	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 “Data set considerations when running CICS with XRF” on page 98 for an explanation of actively and passively shared data sets in an XRF environment.

3 The CICS utility program, DFHTU530, 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 because of the introduction of dynamic allocation for 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 might have to be much larger than your average needs to cater for exceptional cases, 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 are saving space until it becomes necessary to use it. As each extra extent becomes full, VSAM creates another. VSAM continues to create extra extents when 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.

Job **Function**

DFHCOMDS
Deletes and re-creates data sets common to all CICS regions. (See Table 14 on page 97.)

DFHDEFDS
Deletes and re-creates copies of data sets used only by one CICS region. (See Table 15 on page 97.) 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 CICSTS13.CICS.SDFHMSGS 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, CICSTS12.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 OS/390 Installation Guide*.

Table 14. CICS data sets created by the DFHCOMDS job

DFHCSD	CICS region definition data set
SYSIN	SYSIN data set

Table 15. CICS data sets created by the DFHDEFDS 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
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 16. 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 13 on page 94, CICS also uses some MVS data sets. These are listed in Table 17:

Table 17. 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 via the MVS SDUMP macro.

Table 17. MVS data sets used by CICS (continued)

Data set	Owned or used by	Other comments
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 with unwanted SDUMPs that precede ASRA, ASRB, and ASRD abends (after message DFHAP0001). To prevent this, suppress such SDUMPs as described on page 176.

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.

Data set considerations when running CICS with 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 the:

- 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 enqueuing 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 144.

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 DSMShsm 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 DFSMSdfp 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** [5] 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.

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.

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 via 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.

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.

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 10. Defining 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 110. For background information about CICS temporary storage, see the *CICS Application Programming Guide*.

Definition of 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 25 on page 108.

Alternatively, you can run the CICS-supplied job DFHDEFDS (in CICSTS12.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 OS/390 Installation Guide*.

Note: 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(CICSTS13.CICS.CNTL.CICSqualifier.DFHTEMP)-
          RECORDSIZE(4089,4089)           -                      1
          RECORDS(200)                    -
          NONINDEXED                      -
          CONTROLINTERVALSIZE(4096)       -
          SHAREOPTIONS(2 3)               -
          VOLUMES(volid))                 -
        DATA(NAME(CICSTS13.CICS.CNTL.CICSqualifier.DFHTEMP.DATA) -
          UNIQUE)
/*
```

Figure 25. Sample job defining an auxiliary temporary storage data set

Note:

1 The RECORDSIZE value must be 7 bytes less than the CONTROLINTERVALSIZE. See “Space considerations” for information about calculating the control interval size.

Using multiple extents and multiple volumes

The job control statements in Figure 25 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 95.

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
  - 28 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 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 32KB.

  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 may degrade performance. Control interval sizes should be multiples of 512 bytes.

Example: If you use BMS to write a 24 x 80 character screen to temporary storage, the data written occupies 1920 bytes. If you define the temporary storage queue as recoverable, you need 28 bytes for the CICS temporary storage header, giving a total of 1948 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 96. 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=CICSTS13.CICS.applid.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 26.

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

Figure 26. 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.

Approximate storage calculations

The following calculation can be used to determine the approximate total structure size required for a queue pool, given the number of queues and the average data item size.

Storage calculations

```
Item entry size = (170 + (average item size, rounded up to next 256))
                        + 5% extra for control information

Total size   =   200K
                    + (number of large queues x 1K)
                    + (number of items in all queues x item entry size)
```

A large queue is one for which the total size of the data items exceeds 32K. It is stored in a separate list in the structure.

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 that 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 26. Starting up temporary storage servers” on page 367.

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 11. Defining 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*.

Queues used by CICS

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.

CADL CEDA VTAM resource log, indirect to CSSL.

CAFF Transaction Affinities Utility message log, direct to DD name CAFF.

CAIL Autoinstall terminal model resource log, indirect to CSSL.

CCPI CPI Communications message log, indirect to CSSL.

CCSE C/370 error data stream (stderr) log, indirect to CCSO.

CCSI C/370 input data stream (stdin) log. See note.

CCSO C/370 output data stream (stdout) log.

CCZM Messages for CICS classes.

CDBC Database log, indirect to CSSL.

CDUL Dump message log, indirect to CSSL.

CESE Language Environment runtime output.

CMIG Migration log to detect use of EXEC CICS ADDRESS CSA commands.

CPLD PL/I dumps, indirect to CPLI.

CPLI PL/I SYSPRINT output, direct to DD name PLIMSG.

CRDI RDO install log, indirect to CSSL.

CSBA BAM message log.

CSCS Signon/sign-off security log, indirect to CSSL.

CSDH Document Domain message log, indirect to CSSL.

CSDL CEDA command log, indirect to CSSL.

CSFL File allocation message log, indirect to CSSL.

CSKL Transaction and profile resource log, indirect to CSSL.

CSML Signon/sign-off message log, indirect to CSSL.

CSMT Terminal error message and transaction abend message log, indirect to CSSL.

CSNE ZNAC-produced messages log, indirect to CSSL.

CSOO Sockets message log, indirect to CSSL.

CSPL Program resource log, indirect to CSSL.

CSRL Partner resource log, indirect to CSSL.

CSSH Scheduler message log, indirect to CSSL.

CSSL Message log, direct to DD name MSGUSR (all the other general CICS queues are defined as indirect queues to CSSL).

CSTL Terminal I/O error log, indirect to CSSL.

CSZL The queue used for Front End Programming Interface (FEPI) messages. You do not have to define this queue if you do not have FEPI installed.

CSZX The queue used for Front End Programming Interface (FEPI) processing. You do not have to define this queue if you do not have FEPI installed.

CWBO
CICS web messages.

Note: The queue name CCSI has been reserved for the C/370 input data stream (stdin), but any attempt to read from this stream causes EOF to be returned.

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 95.

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$TDWT. This sample program is supplied with the CICS pregenerated system in CICSTS13.CICS.SDFHLOAD, and the assembler source is in CICSTS13.CICS.SDFHSAMP. For programming information about DFH$TDWT, see the *CICS Customization Guide*.

Defining the intrapartition data set

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 27 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” on page 116.

```
//DEFDS     JOB  accounting info,name,MSGCLASS=A
//TDINTRA   EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD   *
       DEFINE CLUSTER -
                ( NAME(CICSTS13.CICS.applid.DFHINTRA)    -
                  RECORDSIZE(1529,1529)            -
                  RECORDS(100)                     -
                  NONINDEXED                       -
                  CONTROLINTERVALSIZE(1536)        -
                  VOL(volid))                      -
              DATA  -
                ( NAME(CICSTS13.CICS.applid.DATA.DFHINTRA))
/*
//
```

Figure 27. 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 27. 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 OS/390 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 27 on page 115 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 95.

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.

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 96. 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  DSNAME=CICSTS13.CICS.applid.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 120 bytes and a record format of V or VB.

You could use the DD statements shown in Figure 28 for the extrapartition data set entries in the sample DCT supplied in CICSTS13.CICS.SDFHLOAD. In these sample DD statements, the LOGUSR queue is defined as a sequential file on disk, and CICSTS13.CICS.LOGUSR is a new data set to be cataloged; the MSGUSR and PLIMSG queues are routed to SYSOUT.

```
//LOGUSR  DD   DSN=CICSTS13.CICS.applid.LOGUSR,DISP=(NEW,KEEP),
//        DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),
//        VOL=SER=volid,UNIT=3380,SPACE=(CYL,2)
//MSGUSR  DD   SYSOUT=A
//PLIMSG  DD   SYSOUT=A
```

Figure 28. Sample JCL to define transient data extrapartition data sets

Note: Change the space allocation given in this sample job stream to suit your own installation's needs.

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 29.) 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 29 in a startup job stream for an alternate CICS region.

```
//DFHCXRF  DD SYSOUT=*
```

or

```
//DFHCXRF   DD  DSN=CICSTS13.CICS.applid.DFHCXRF,DISP=(NEW,KEEP),
//                              DCB=(DSORG=PS,RECFM=V,BLKSIZE=136),
//                              VOL=SER=volid,UNIT=3380,SPACE=(TRK,5)
```

Figure 29. 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 118.)

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 12. Defining 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

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.

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(CICSLOGS)
        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(J10TO19) 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 OS/390 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 Recovery and Restart 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 log 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.

Journal naming

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 18.

Table 18. 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 DFHJ*nn* 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 19 for an example of this.

Table 19. 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 19. 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 DFHJ*nn* 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 DFHJ*nn* 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*.DFHJ*nn*.

Table 20 shows an example of a user journal name table entry.

Table 20. 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.

JOURNALMODEL definitions

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
After you discard the journal name, and all files that are using the log stream for autojournaling are closed.

Forward recovery logs (excluding those defined in the ICF catalog**)**
After you discard the journal name, and 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 30 on page 128).

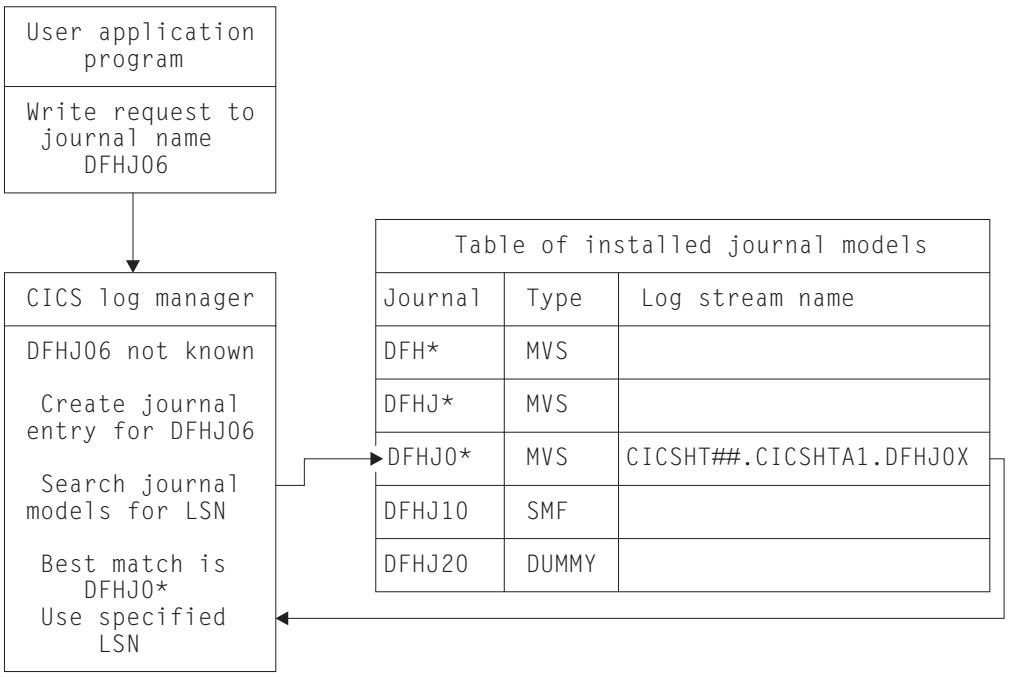


Figure 30. Looking for a journal model that matches a journal name. CICS searches the journal model table to find the log stream name that corresponds to the journal name, using a "best-match" algorithm.

How CICS logs and journals map on to 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. See Figure 31 on page 130 and Figure 32 on page 132 for diagrams of the mapping process.

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. See Figure 31 on page 130 for a graphical representation of the system log mapping process.

On an initial start, 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.DFHSHUNT

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 DFHSHUNT 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 the following default names:

- sysname.DFHLOG.MODEL
- sysname.DFHSHUNT.MODEL

where ‘sysname’ is the name of the MVS system on which CICS is running. Once these log streams have been created, CICS then connects to them.

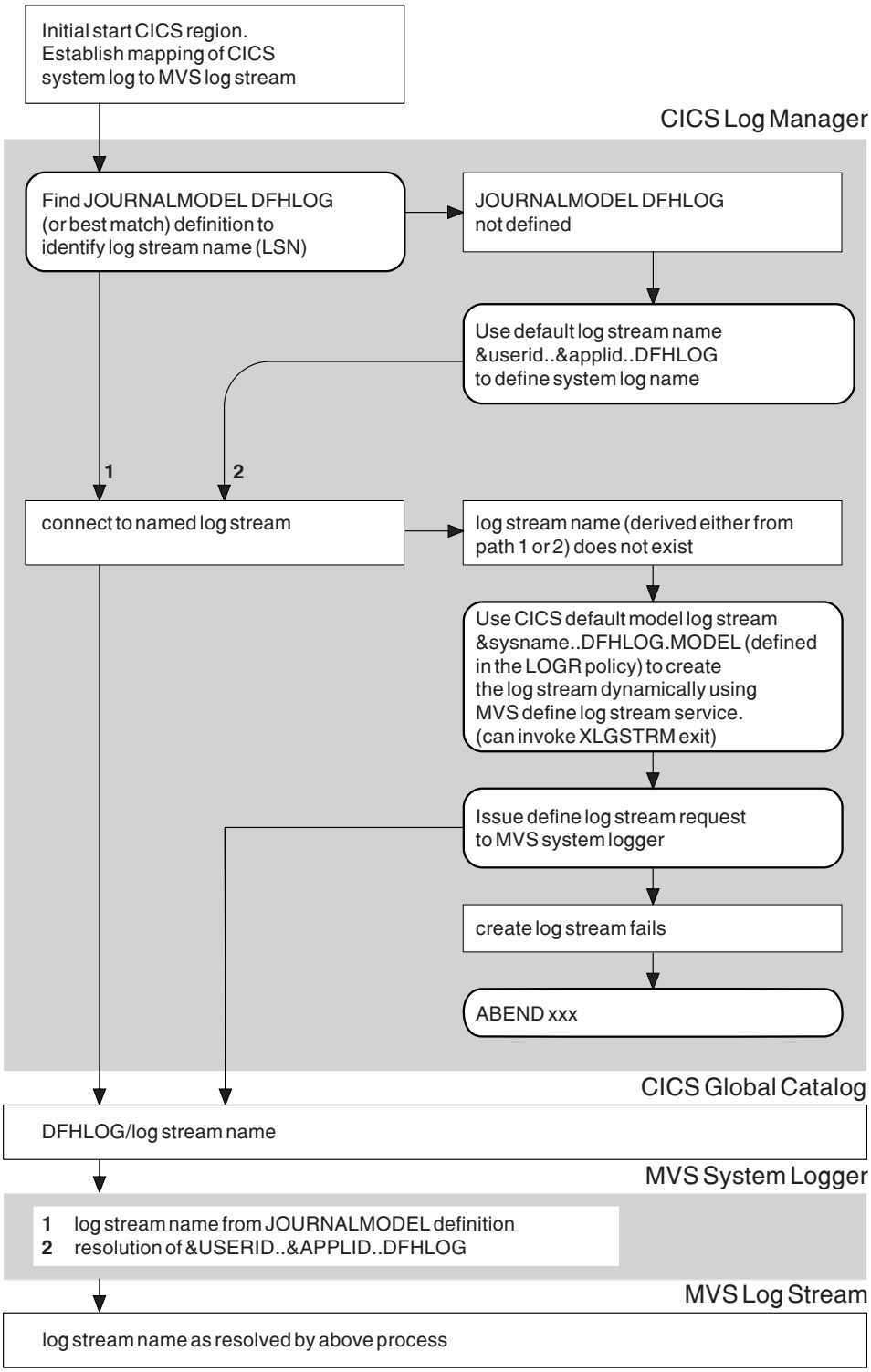


Figure 31. How CICS maps the system log (DFHLOG) to a log stream name (LSN) during an INITIAL start. CICS uses the same process for the secondary system log, DFHSHUNT.

Mapping of general log streams

CICS uses the default log stream names unless you provide a JOURNALMODEL resource definition for the journal or log. See Figure 32 on page 132 for a graphical representation of the mapping process for general logs.

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 following 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 following 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.

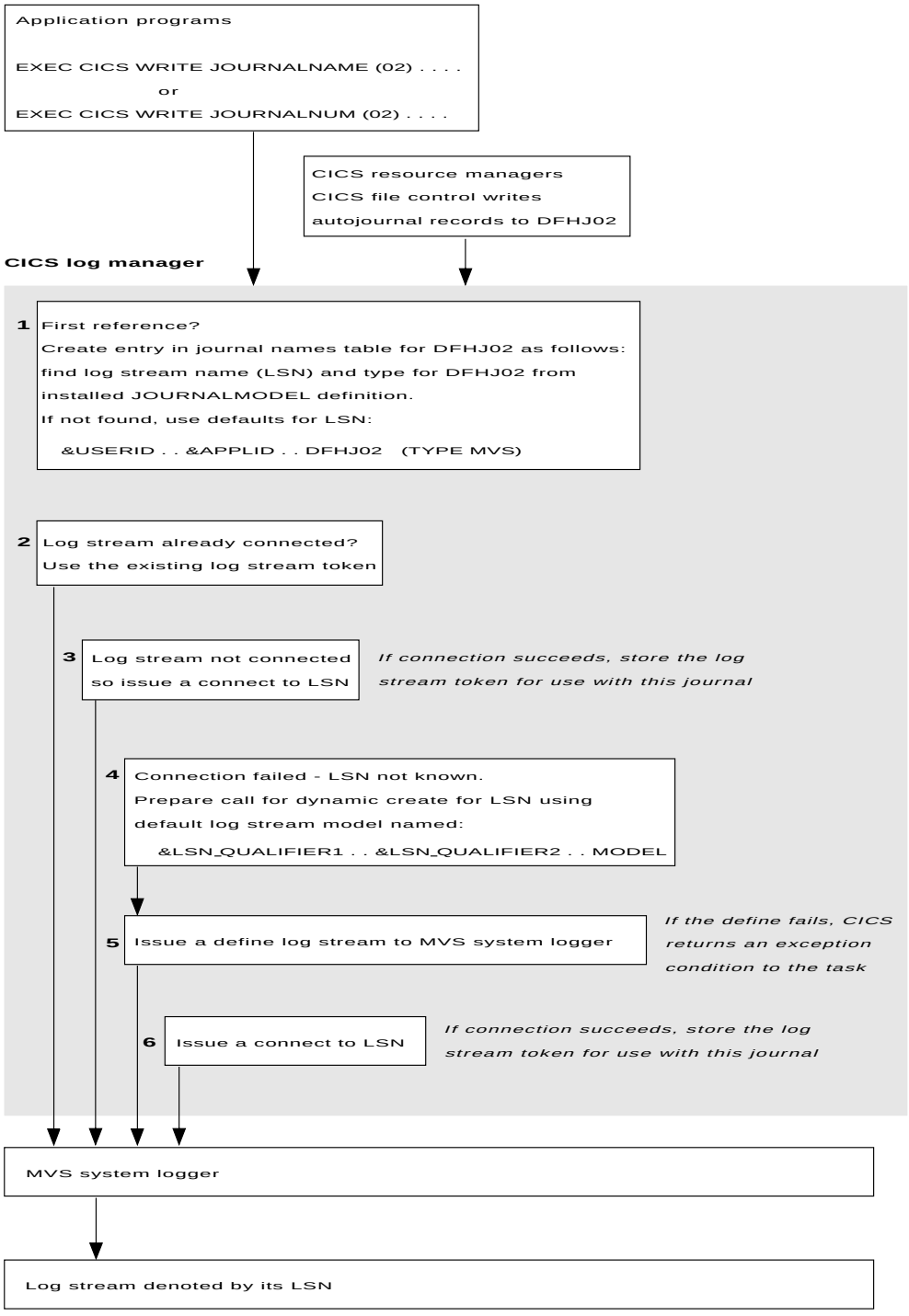


Figure 32. How a CICS journal is mapped to its log stream name (LSN). The name is DFHJ02, used here for user journaling and file control autojournaling.

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 13. Defining 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 OS/390 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 “Summary of steps to create 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 OS/390 Release 3. 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.

Summary of steps to create 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 192.
3. Decide whether the CSD is to be eligible for backup-while-open (BWO[6]). If so, you require the following components of DFSMS 1.2 or later:
   - DFSMShsm
   - DFSMSdss

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

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.
- 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 CSDxxxxx system initialization parameters (see “File processing attributes for the CSD” on page 139).
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 “RDO command logs” on page 153 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 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)
Program	416 bytes

Resource	Definition record size (maximum)
Transaction	192 bytes
Profile	156 bytes
Partition set	146 bytes
Map set	146 bytes
Terminal	313 bytes
Typeterm	336 bytes
Connection	184 bytes
Session	260 bytes
File	264 bytes
LSR pool	288 bytes
Transient data queue	192 bytes
Journalmodel	166 bytes
DB2CONN	192 bytes
DB2ENTRY	128 bytes
DB2TRAN	128 bytes
Doctemplate	160 bytes
Enqmodel	352 bytes
Partner	100 bytes
Processtype	128 bytes
Requestmodel	224 bytes
TCPIPSERVICE	192 bytes
Tranclass	96 bytes
Tsmodel	128 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 33 on page 138.)

Defining and initializing the CICS system definition

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 33 on page 138.)

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
//DEFCSD   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SYSIN    DD   *
   DEFINE CLUSTER -
            (NAME(CICSTS13.CICS.applid.DFHCSD) -
            VOLUMES(volid)               -
            KEYS(22 0)                   -                          1
            INDEXED                      -
            RECORDS(n1 n2)               -
            RECORDSIZE(120 500)          -                          2
            FREESPACE(10 10)             -
            SHAREOPTIONS(2)              -                          3
            LOG(ALL)                     -                          4
            LOGSTREAMID(CICSTS13.CICS.CSD.FWDRECOV))  -                 4
            BWO(NO)                                                  4

          DATA                           -
            (NAME(CICSTS13.CICS.applid.DFHCSD.DATA)   -
            CONTROLINTERVALSIZE(8192))          -
          INDEX                          -
            (NAME(CICSTS13.CICS.applid.DFHCSD.INDEX))
/*
//INIT     EXEC PGM=DFHCSDUP,REGION=300K
//STEPLIB  DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=CICSTS13.CICS.applid.DFHCSD,DISP=SHR                 5
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN    DD *
           INITIALIZE
           LIST ALL OBJECTS
/*
//
```

Figure 33. Sample job to define and initialize the CSD

Notes:

1 The key length is 22 bytes, and the KEYS parameter must be coded as shown.

2 The average record size of 120 bytes is calculated for a CSD 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 CSD, 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 disk space” on page 136.

3 Code the SHAREOPTIONS parameter as shown.

4 If you are using DFSMS 1.3, you can specify the recovery attributes for the CSD in the ICF catalog instead of using the CSD system initialization parameters. If you decide to use the CSD 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.

File processing attributes for the CSD

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

CSDACC
The type of access allowed.

CSDBKUP
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.

CSDBUFND
The number of buffers for CSD data. Ignored if CSDRLS=YES.

CSDBUFNI
The number of buffers for the CSD index. Ignored if CSDRLS=YES.

CSDDISP
The disposition of the CSD data set.

CSDDSN
The JCL data set name (DSNAME) of the CSD.

CSDFRLOG
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.

CSDINTEG
The level of read integrity to be used for a CSD accessed in RLS-mode.

CSDJID
An identifier for automatic journaling.

CSDLSRNO
A VSAM local shared resource pool. Ignored if CSDRLS=YES.

CSDRECOV
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 ICG catalog specifies NONE, UNDO, or ALL, the LOG parameter overrides CSDRECOV.

CSDRLS
Whether the CSD is accessed in RLS or non-RLS mode.

CSDSTRNO
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 21. CICS system initialization parameters” on page 215.

Sharing and availability of 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 143.

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 “sample job stream” on page 138.
- 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 143

page 143. For more information about shared CSD access across several MVS images, see “Sharing a CSD in a multi-MVS environment (non-RLS)” on page 144. For information about sharing the CSD between different releases of CICS, see “Sharing the CSD between different releases of CICS” on page 145.

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 146.

For information about the system initialization parameters for controlling access to the CSD, see “File processing attributes for the CSD” on page 139.

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 “sample job stream” on page 138. 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 141.

  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 148 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 150 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 system initialization parameters allow the following operations on the CSD:

CSDACC operand
: **Operations**

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 enqueuing (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 21 on page 145.

Table 21. 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 21 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 CED*x* 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 CED*x* 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 CED*x* 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 can 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 OS/390 Release 3 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 OS/390 Release 3 level, some of the IBM groups referenced by your group list are deleted, and the contents transferred to one of the compatibility groups, DFHCOMP*x*. 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 OS/390 Release 3, 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 143 , 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 150.

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 CED*x* 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 CED*x* 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 CED*x* 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 CED*x* 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 CSD*xxxxx* system initialization parameters (see “File processing attributes for the CSD” on page 139) except for the following:

CSDBKUP
: You specify backup-while-open support for the CSD using the VSAM BWO parameter in the ICF catalog.

CSDBUFND
: Ignored.

CSDBUFNI
: Ignored.

CSDFRLOG
: You specify the forward recovery log stream for the CSD using the VSAM LOGSTREAMID parameter in the ICF catalog.

CSDINTEG
: You specify read integrity for RDO transactions (CED*x*) using this system initialization parameter.

CSDLSRNO
: Ignored.

CSDRECOV
: 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.

CSDSTRNO
: 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 CED*x* 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 101.

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 "RDO command logs" on page 153 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 22 and Table 23 summarize their effects when the SIT is assembled and during CICS override processing, respectively.

Table 22. 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 23. 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 23. 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 and availability of the CSD in non-RLS mode” on page 140. 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.

RDO command logs

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:

CADL 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.

CAIL Logs autoinstall terminal model entries installed in the TCT, and entries deleted from the TCT.

CRDI Logs installed resource definitions of programs, transactions, mapsets, profiles, partition sets, files, and LSR pools.

CSDL Logs RDO commands that affect the CSD.

CSFL 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.

CSKL 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.

CSPL 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.

CSRL 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 34 on page 155. 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 34. 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=CICSTS13.CICS.applid.DFHCSD,DISP=SHR
  ```

  You usually need the CSD DD statement to include DISP=SHR. (See “Sharing and availability of the CSD in non-RLS mode” on page 140.)

  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 18. Defining user files” on page 189. For information about the parameters that you can code for the CSD in the SIT, see “Chapter 21. CICS system initialization parameters” on page 215.

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 OS/390 Migration Guide*.

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*.

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 99.)

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 “File processing attributes for the CSD” on page 139.

For more information about using the CSD as a recoverable file, see “Planning for backup and recovery” on page 150.

Chapter 14. Defining and using catalog data sets

This chapter 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 chapter, 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 OS/390 Release 3.
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 325.

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
  - Sessions and connections, for communication with other CICS regions
  - Terminals, including any that are autoinstalled
  - Transactions
  - Transaction classes
  - Transient data queues
- 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 325.

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 35 on page 160 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 OS/390 Installation Guide*.

```
//GLOCAT   JOB accounting info,,CLASS=A
//DEFGCD   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
      DEFINE CLUSTER -
               (NAME(CICSTS13.CICS.applid.DFHGCD) -          1
               INDEXED                        -
               CYLINDERS(n1 n2))              -                    2
               FREESPACE(10 10)               -
               SHAREOPTIONS(2)                -
               REUSE                          -                    3
               VOLUMES(volid))                -
             DATA                             -                    4
               (NAME(CICSTS13.CICS.applid.DFHGCD.DATA)  -
               CONTROLINTERVALSIZE(8192)      -                    5
               KEYS(28 0))                    -
             INDEX                            -
               (NAME(CICSTS13.CICS.applid.DFHGCD.INDEX) )
/*
//INITGCD  EXEC PGM=DFHRMUTL,REGION=1M                             6
//STEPLIB  DD DSNAME=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD   DD DSNAME=CICSTS13.CICS.applid.DFHGCD,DISP=OLD
//SYSIN    DD *
SET_AUTO_START=AUTOINIT
/*
                                                                   7
```

Figure 35. 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 This job does not specify a RECORDSIZE value for the global catalog, which therefore defaults to using an average and maximum record size of 4089 bytes, RECORDSIZE(4089 4089). If your maximum record size is greater than 4089, you must add a RECORDSIZE parameter to the sample job to specify your own value. For information about record sizes, see Table 24 on page 163.

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 35 on page 160, the absolute minimum value of BUFSP is calculated as follows:

```
BUFND = (STRNO + 1)
BUFNI = STRNO
BUFSP = 33 * 8192 (BUFND * CI size) + 32 * 1024  (BUFNI * CI size) =
        303104 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 35 on page 160.

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 36 on page 163 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 “Classes of start and restart” on page 325.

```
//RMUTL    EXEC PGM=DFHRMUTL,REGION=1M
//STEPLIB  DD DSNAME=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFHGCD   DD DSNAME=CICSTS13.CICS.applid.DFHGCD,DISP=OLD
//NEWGCD   DD DSNAME=CICSTS13.CICS.applid.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=CICSTS13.CICS.applid.DFHGCD,DISP=OLD
//NEWGCD   DD DSNAME=CICSTS13.CICS.applid.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 36. 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 24.

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.

Table 24. Sizes for entries in the global catalog

Installed definition, table entry, or control block	***Number of bytes per entry***
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

Table 24. Sizes for entries in the global catalog (continued)

Installed definition, table entry, or control block	***Number of bytes per entry***
Installed remote queue definition	84 bytes
Installed TRANSACTION definitions (without TPNAME)	112 bytes
Installed TRANSACTION definitions (with TPNAMEor 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 **1**	1156 bytes
Data set names (JCL or dynamically allocated) **2**	52 bytes
Data set name blocks	115 bytes
File control recovery blocks **3**	97 bytes
Terminal control table entry (TCT)	
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 **4**	582 bytes
Installed model TERMINAL definitions **4**	582 bytes
Deferred work element (DWE) **5**	80 bytes
Installed journal	88 bytes
Installed journalmodel	80 bytes
Recovery manager remote names	106 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 via 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 35 on page 160, the data definition statement for the CICS execution is:

```
//DFHGCD  DD  DSN=CICSTS13.CICS.applid.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 24. 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.

The local catalog

CICS Transaction Server for OS/390 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 “Classes of start and restart” on page 325.

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

  DFHAP
  Application domain.

  DFHBA
  Business application manager.

  DFHCC
  CICS local catalog domain.

  DFHDD
  Directory manager domain.

  DFHDH
  Document handler domain.

  DFHDM
  Domain manager domain.

  DFHDS
  Dispatcher domain

  DFHDU
  Dump domain.

  DFHEM
  Event manager domain

  DFHGC
  CICS global catalog domain.

  DFHKE
  Kernel domain.

  DFHLD
  Loader domain.

  DFHLG
  Log manager domain.

  DFHLM
  Lock manager domain.

  DFHME
  Message domain.

  DFHMN
  Monitoring domain.

  DFHNQ
  Enqueue manager domain

  DFHPA
  System initialization parameter domain.

DFHPG
Program manager domain.

DFHRM
Recovery manager domain

DFHRX
RRMS domain

DFHSH
Scheduler services domain

DFHSM
Storage manager domain.

DFHSO
Sockets domain.

DFHST
Statistics domain.

DFHTI Timer domain.

DFHTR
Trace domain.

DFHUS
User domain.

DFHWB
Web domain

DFHXM
Transaction manager domain.

DFHXS
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 37 on page 169.

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 37 on page 169. 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 OS/390 Installation Guide*.

```
//LOCAT    JOB accounting info,,CLASS=A
//DEFLCD   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
//*
         DEFINE CLUSTER -
                (NAME( CICSTS13.CICS.applid.DFHLCD) -                  1
                 INDEXED                          -
                 RECORDS( 200 10 )                -                    2
                 FREESPACE(10 10)                 -
                 SHAREOPTIONS( 2 )                -
                 REUSE                            -
                 VOLUMES( volid ))                -
           DATA                                   -
                (NAME( CICSTS13.CICS.applid.DFHLCD.DATA )   -
                 KEYS( 28 0 )                     -
                 RECORDSIZE( 45 124 )             -                    3
                 CONTROLINTERVALSIZE( 2048 ))  -
            INDEX (NAME( CICSTS13.CICS.applid.DFHLCD.INDEX ) )
/*
//*************************************************************
//INITLCD EXEC PGM=DFHCCUTL
//*
//*            INITIALIZE THE CICS LOCAL CATALOG
//*
//STEPLIB   DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//DFHLCD    DD DSN=CICSTS13.CICS.applid.DFHLCD,DISP=SHR
//*
//
```

Figure 37. 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( CICSTS13.CICS.DBDCCIC1.DFHLCD)
  ⋮
DEFINE CLUSTER -
       (NAME( CICSTS13.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 37, the data definition statement for the CICS execution is:

```
//DFHLCD  DD  DSN=CICSTS13.CICS.applid.DFHLCD,DISP=OLD
```

Chapter 15. Defining and using auxiliary trace data sets

This chapter 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*.

Defining auxiliary trace data sets

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 DFHTU530 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

AUXTR
Switches auxiliary trace on or off at CICS startup.

AUXTRSW
Specifies automatic switching for auxiliary trace data sets when full.

INTTR Switches internal trace on or off at CICS startup.

GTFTR
Specifies whether CICS is to use GTF as a destination for CICS trace data.

SPCTR
Specifies the level of special tracing.

SPCTRxx
Specifies the level of special tracing for the “xx” component.

STNTR
Specifies the level of CICS standard tracing.

STNTRxx
Specifies the level of standard tracing for the “xx” component.

SYSTR
Switches the system master trace flag on or off at CICS startup.

TRTABSZ
Defines the size of the CICS internal trace table.

USERTR
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 21. CICS system initialization parameters” on page 215.

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 the *CICS Supplied Transactions* manual.

Job control statements to allocate auxiliary trace data sets

If you are defining auxiliary trace data sets on disk, you can use the job shown in Figure 38 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 OS/390 Installation Guide*.

```
//DEFTRCDS JOB (accounting information),
//             MSGCLASS=A,MSGLEVEL=(1,1),
//             CLASS=A,NOTIFY=userid
//*******************************************************************
//*            Create auxiliary trace data sets
//*******************************************************************
//ALLOCDS  EXEC PGM=IEFBR14
//DFHAUXT  DD  DSN=CICSTS13.CICS.applid.DFHAUXT,UNIT=3380,VOL=SER=volid,
//         DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),  1
//              SPACE=(CYL,(5,1))                                   2
//DFHBUXT  DD  DSN=CICSTS13.CICS.applid.DFHBUXT,UNIT=3380,VOL=SER=volid,
//         DISP=(NEW,CATLG),DCB=(BLKSIZE=4096,RECFM=F,LRECL=4096),  1
//              SPACE=(CYL,(5,1))                                   2
//
```

Figure 38. 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 38, you can define them to CICS in the startup job stream with the following DD statements:

```
//DFHAUXT  DD  DSN=CICSTS13.CICS.applid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
//DFHBUXT  DD  DSN=CICSTS13.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.

DISP=SHR allows the simultaneous processing of a data set by the DFHTU530 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=CICSTS13.CICS.applid.DFHAUXT,UNIT=3400,VOL=SER=volid,
//         DISP=(NEW,KEEP),LABEL=(,NL)
//DFHBUXT  DD  DSN=CICSTS13.CICS.applid.DFHBUXT,UNIT=3400,VOL=SER=volid,
//         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 the *CICS Supplied Transactions* manual.

Trace utility program (DFHTU530)

If you write trace entries to CICS auxiliary trace data sets you can use the trace utility program, DFHTU530, 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 DFHTU530, see the *CICS Operations and Utilities Guide*.

Chapter 16. 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 176 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 via 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 353.

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* manual 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 DFHDU530 dump utility program. For information about the DFHDU530 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 DFHDU530 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 DFHDU530 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 179 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. XDUCLSE, 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 21. CICS system initialization parameters” on page 215.

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 OS/390 Installation Guide*.

Alternatively, you can use the sample data definition statements in Figure 39 on page 179 to allocate and catalog dump data sets on disk.

```
//DFHDMPA  DD  DSN=CICSTS13.CICS.applid.DFHDMPA,DISP=(NEW,CATLG),
//             UNIT=3380,VOL=SER=volid,SPACE=(CYL,(5,1))
//DFHDMPB  DD  DSN=CICSTS13.CICS.applid.DFHDMPB,DISP=(NEW,CATLG),
//             UNIT=3380,VOL=SER=volid,SPACE=(CYL,(5,1))
```

Figure 39. 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 DFHDU530 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 39.

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:

```
//DFHDMPA  DD  DSN=CICSTS13.CICS.applid.DFHDMPA,DISP=SHR
//DFHDMPB  DD  DSN=CICSTS13.CICS.applid.DFHDMPB,DISP=SHR
```

DISP=SHR enables each data set, if held on disk, to be processed by the DFHDU530 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:

```
//DFHDMPA  DD  DSN=CICSTS13.CICS.applid.DFHDMPA,UNIT=3400,VOL=SER=volid1,
//         DISP=(NEW,KEEP),LABEL=(,NL)
//DFHDMPB  DD  DSN=CICSTS13.CICS.applid.DFHDMPB,UNIT=3400,VOL=SER=volid2,
//         DISP=(NEW,KEEP),LABEL=(,NL)
```

Chapter 17. 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 40 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 OS/390 Installation Guide*.

```
//CICSCTL   JOB   'accounting info',name,MSGCLASS=A
//XRCTL     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
       DEFINE CLUSTER -
                (NAME(CICSTS13.CICS.applid.DFHXRCTL) -
                 RECORDSIZE(4089 4089)         -                  1
                 CONTROLINTERVALSIZE(4096)     -                  2
                 RECORDS(4)                    -
                 NIXD                          -
                 SHAREOPTIONS(3,3)             -                  3
                 VOLUMES(volid1))              -
              DATA                             -                  4
                (NAME(CICSTS13.CICS.applid.DFHXRCTL.DATA))
/*
//
```

Figure 40. 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=CICSTS13.CICS.applid.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 41 on page 184 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 OS/390 Installation Guide*.

If you use the sample JCL in Figure 41 on page 184, 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.

```
//CICSMSG   JOB   'accounting info',name,MSGCLASS=A
//XRMSG     EXEC  PGM=IDCAMS
//DDNAME2   DD    DISP=OLD,UNIT=3380,VOL=SER=volid2
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
       DEFINE CLUSTER -
                (NAME(CICSTS13.CICS.applid.DFHXRMSG) -
                 RECORDSIZE(4089 4089)         -                  1
                 CONTROLINTERVALSIZE(4096)     -                  2
                 RECORDS(1500)                 -
                 NIXD                          -
                 SHAREOPTIONS(3,3)             -                  3
                 VOL(volid2)                   -
                 FILE(DDNAME2))                -
              DATA                             -                  4
                (NAME(CICSTS13.CICS.applid.DFHXRMSG.DATA))
/*
//
```

Figure 41. 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 25 lists the sizes of the various messages sent to the data set.

Table 25. Sizes of messages sent to the XRF message data set

Type of TCT entry	***Bytes per install***
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
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 26. 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=CICSTS13.CICS.applid.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 18. 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 95.

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 192 for further information.

Each of the above methods is discussed under the following topics:

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 191.

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 101.

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 via 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 via 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 42 on page 193.

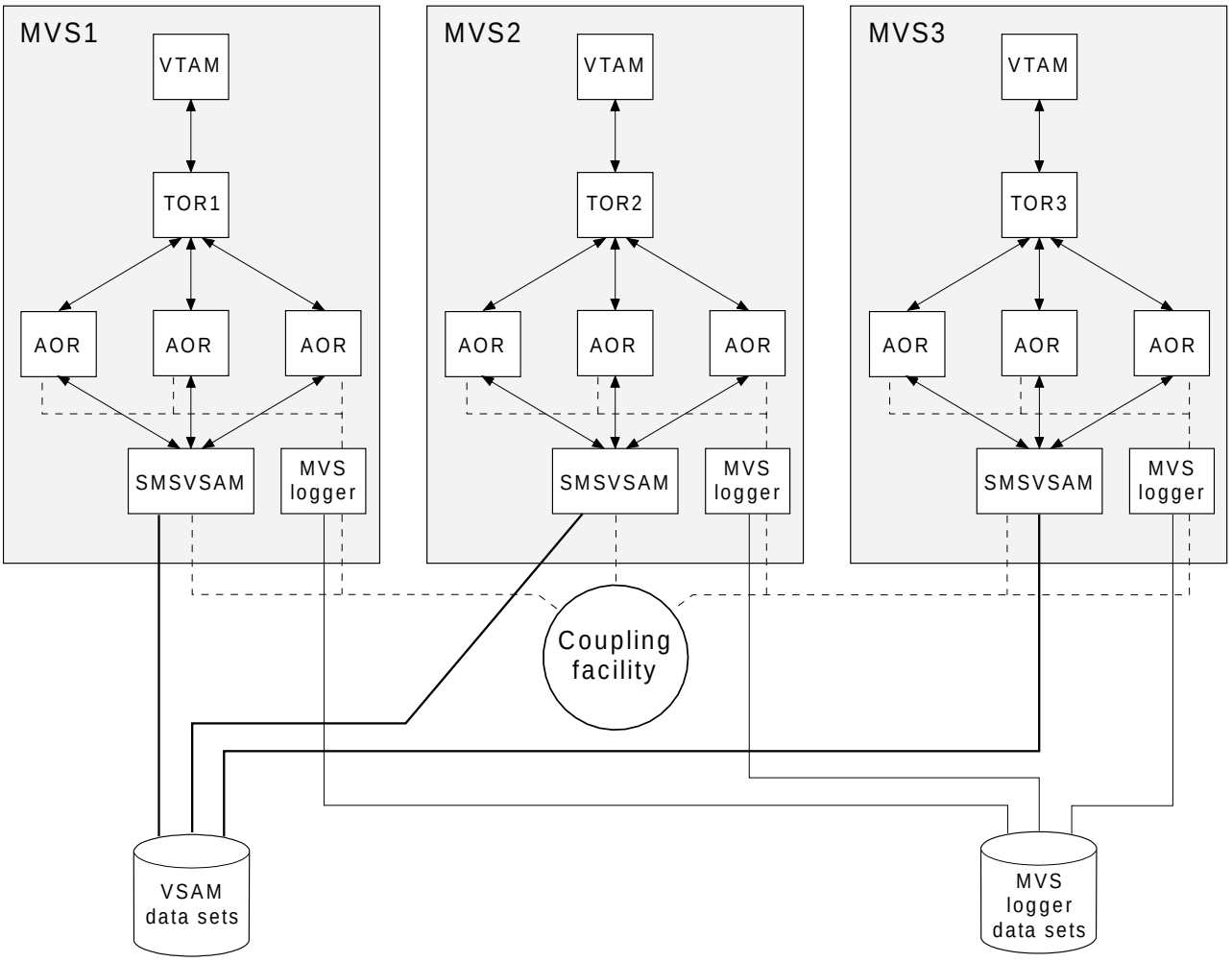


Figure 42. Diagram illustrating a parallel sysplex with RLS. This view of RLS shows multiple CICS regions using VSAM RLS, through the services of an SMSVSAM server in each MVS image.

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) data is not supported in RLS access mode. However, path access to data is supported.
- VSAM catalogs and VVDS data sets are not supported.

For details of all the steps necessary to set up support for VSAM RLS, see the *CICS Transaction Server for OS/390 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 43.

Notes:

```
//BDAM EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=CICSTS12.bdam.user.file.init,DISP=SHR        1
//SYSUT2   DD DSN=CICSTS12.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 43. Sample JCL to create and load a BDAM data set

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 44, 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 44. 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** via the CSD, and BDAM files **only** via 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 DSNAME 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=CICSTS13.CICS.vsam.user.file,DISP=OLD
//BDAMFILE  DD   DSN=CICSTS13.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 DSNAME and DISP file resource definition parameters

You can define a data set to CICS by specifying the DSNAME 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 DSNAME 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 DSNAME 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 DSNAME and DISP parameters, see SRT system recovery table and 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) DSNAME(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 Supplied Transactions* manual.

Dynamic allocation in an application program

You can assign the data set name dynamically from an application program by using the command:

```
EXEC CICS SET FILE(filename) DSNAME(datasetname) SHARE|OLD
```

The data set is then dynamically allocated in the same way as if you used the CEMT master terminal command, but only if the file status is CLOSED, and also DISABLED or UNENABLED. The file must be closed when you issue a command to change file attributes.

For programming information about the EXEC CICS SET command, see the *CICS System Programming Reference* manual.

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 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 CEDA DEFINE FILE command. When a table is opened, CICS builds it by extracting data from the table's 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* manual. CICS supports two types of data tables:

- **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 199.

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 200.

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 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.

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 384 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 27. Starting a coupling facility data table server” on page 381 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 adminstrative 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 45.

```
STRUCTURE  NAME(DFHCFLS_PRODCFT1)
   SIZE(1000)
   INITSIZE(500)
   PREFLIST(FACIL01,FACIL02)
```

Figure 45. 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=`*`policyname`*`,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 you dynamically increase the size of a list structure, 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

Use the following calculation to determine the approximate total structure size required for a coupling facility data table pool, given the number of tables and records, and the average record size.

Storage calculations

```
Data entry size = (170 + (average record data size[1]))
                  + 5% extra for control information

[1] Average record data size must have a 2-byte prefix added
       and be rounded up to a multiple of 256 bytes.

Total size  =  200KB
               + (number of tables x 1KB)
               + (number of records in all tables x 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.

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 391 and “Avoiding structure full conditions” on page 392.

Chapter 19. 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 46. 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 CICSTS13.CICS.DFHDBFK
  SET MAXCC=0
  DEFINE CLUSTER (                              -
                NAME( CICSTS13.CICS.DFHDBFK )  -
                INDEXED                         -
                RECORDS(100 20)                 -
                KEYS(22,0)                      -
                RECORDSIZE(1428 1428)           -
                )                               -
  INDEX         (                               -
                NAME( CICSTS13.CICS.DFHDBFK.INDEX )     -
                CONTROLINTERVALSIZE(512)        -
                )                               -
  DATA          (                               -
                NAME( CICSTS13.CICS.DFHDBFK.DATA )      -
                CONTROLINTERVALSIZE(2048)       -
                )
/*
//*     The next two job steps are optional.
//*
//DBFKINID EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE  CICSTS13.CICS.DBFKINIT
/*
//DBFKINIF EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSN=CICSTS13.CICS.DBFKINIT,DISP=(NEW,CATLG),
// UNIT=dbfkunit,VOL=SER=dbfkvol,SPACE=(TRK,(1,1)),
// DCB=(RECFM=FB,LRECL=40,BLKSIZE=6160)
//*  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)
/*
```

Figure 46. Sample job to define and initialize the DFHDBFK data set (Part 1 of 2)

```
//DBFKLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYS01    DD DSN=CICSTS13.CICS.DBFKINIT,DISP=SHR
//DFHDBFK  DD DSN=CICSTS13.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 46. 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 46 on page 207, the data definition statement for the CICS execution is as follows:

```
//DFHDBFK  DD DSN=CICSTS13.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 *CICS IMS Database Control Guide* for details). Leading or embedded blanks are not allowed, but trailing blanks are acceptable. **Note:** 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.) **Note:** 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 20. Defining the CMAC messages data set

This chapter describes the VSAM key-sequenced data set (KSDS) called DFHCMACD. DFHCMACD is used by the CMAC transaction to provide online descriptions of the CICS messages and codes.

You can create the DFHCMACD data set and load it with the CICS-supplied messages and codes data by running the DFHCMACI job. Some IBM-supplied service may include changes to CICS messages and codes, and associated changes to the DFHCMACD data set. You can apply such service changes to the DFHCMACD data set by running the DFHCMACU job.

For more information about the DFHCMACI and DFHCMACU jobs, see the *CICS Transaction Server for OS/390 Installation Guide*.

Notes:

1. The DFHCMACD 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 DFHCMAC. 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 DSNAME 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 DFHCMAC 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 DFHCMAC 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 DFHCMACD data set should be defined and loaded as a VSAM key sequenced data set (KSDS). The sample job in Figure 47 on page 212 shows you how to do this.

Note: You can define and load the DFHCMACD data set by running the DFHCMACI job.

```
//CMACJOB  JOB  'accounting information',name,MSGCLASS=A
//CMACDEF  EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYSIN    DD *
  DELETE CICSTS13.CICS.DFHCMACD
  SET MAXCC=0
  DEFINE CLUSTER (                              -
                NAME( CICSTS13.CICS.DFHCMACD )      -
                CYL(2,1)                        -
                KEYS( 9 0 )                     -
                INDEXED                         -
                VOLUME ( cmacvol)               -
                RECORDSIZE( 8192 32666 )        -
                FREESPACE( 5 5 )                -
                SHAREOPTIONS( 2 )               -
                )                               -
  INDEX         (                               -
                NAME( CICSTS13.CICS.DFHCMACD.INDEX )    -
                )                               -
  DATA          (                               -
                NAME( CICSTS13.CICS.DFHCMACD.DATA )     -
                )
/*
//CMACLOAD EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*
//AMSDUMP  DD SYSOUT=*
//SYS01    DD DSN=CICSTS13.CICS.SDFHMSGS(DFHCMACD),DISP=SHR
//DFHCMACD DD DSN=CICSTS13.CICS.DFHCMACD,DISP=SHR
//SYSIN    DD *
  REPRO INFILE (SYS01)                    -
        OUTFILE (DFHCMACD)
/*
//
```

where `cmacvol` is the volume on which the DFHCMACD data set is to be created.

Figure 47. 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 47, the data definition statement for the CICS execution is:

```
//DFHCMACD  DD DSN=CICSTS13.CICS.DFHCMACD,DISP=SHR
```

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 21. CICS system initialization parameters” on page 215 describes the CICS system initialization parameters.
- “Chapter 22. Processing system initialization parameters” on page 319 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 24. CICS startup” on page 337 describes a sample startup job stream, and a sample procedure for use as a started task.

Chapter 21. 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.

Specifying system initialization parameters

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 27 on page 216. 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 236 for an example of such a restriction.

There are some other CICS system initialization parameters (and options of the parameters in Table 27 on page 216) that you cannot define in the DFHSIT macro. (See "Initialization parameters that cannot be coded in the DFHSIT macro" on page 226.) The parameters that you cannot define in the DFHSIT macro are shown in Figure 54 on page 227.

For a list of all the system initialization keywords grouped by their functional area, see "Appendix. System initialization parameters grouped by functional area" on page 417. This ensures that you do not miss coding an important parameter relating to a particular CICS function. For details of how to code a parameter, you still have to refer to the parameter descriptions that are listed alphabetically in this chapter.

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 OS/390 Release 3 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 "DFHSIT, the default system initialization table" on page 220.

The DFHSIT macro parameters

Table 27. The DFHSIT macro parameters

	DFHSIT	[TYPE={**CSECT**|DSECT}] [,ADI={**30**|number}] [,AICONS={**NO**|YES|AUTO}] [,AIEXIT={**DFHZATDX**|DFHZATDY|name}] [,AILDELAY={**0**|hhmmss}] [,AIQMAX={**100**|number}] [,AIRDELAY={**700**|hhmmss}] [,AKPFREQ={**4000**|number}] [,APPLID=({**DBDCCICS**|name1}[,name2])] [,AUTCONN={**0**|hhmmss}] [,AUXTR={**OFF**|ON}] [,AUXTRSW={**NO**|ALL|NEXT}] [,BMS=({MINIMUM|STANDARD|**FULL**}[,COLD][[,{**UNALIGN**|ALIGN}][,{**DDS**|NODDS}])] [,CDSASZE={**0K**|number}] [,CICSSVC={**216**|number}] [,CLSDSTP={**NOTIFY**|NONOTIFY}] [,CLT=xx] [,CMDPROT={**YES**|NO}] [CMDSEC={**ASIS**|ALWAYS} [,CONFDATA={**SHOW**|HIDETC}] [,CONFTXT={**NO**|YES}] [,CSDACC={**READWRITE**|READONLY}] [,CSDBKUP={**STATIC**|DYNAMIC}] [,CSDBUFND=number] [,CSDBUFNI=number] [,CSDDISP={OLD|SHR}] [,CSDDSN={name}] [,CSDFRLOG=number] [,CSDINTEG={**UNCOMMITTED**|CONSISTENT| REPEATABLE}] [,CSDJID={**NO**|number}] [,CSDLSRNO={**1**|number|NONE|NO}] [,CSDRECOV={**NONE**|ALL|BACKOUTONLY}] [,CSDRLS={**NO**|YES}]

Table 27. The DFHSIT macro parameters (continued)

		[,CSDSTRNO={**2**|number}] [,CWAKEY={**USER**|CICS}] [,DAE={**NO**|YES}] [,DATFORM={**MMDDYY**|DDMMYY|YYMMDD}] [,DB2CONN={**NO**|YES}] [,DBCTLCON={**NO**|YES}] [,DCT=({YES|xx|**NO**})] [,DFLTUSER={**CICSUSER**|userid}] [,DIP={**NO**|YES}] [,DISMACP={**YES**|NO}] [,DOCCODEPAGE={**037**|codepage}] [,DSALIM={**5M**|number}] [,DSHIPIDL={**020000**|hhmmss}] [,DSHIPINT={**120000**|hhmmss}] [,DSRTPGM={**NONE**|DFHDSRP|program-name|EYU9XLOP}] [,DTRPGM={**DFHDYP**|program-name}] [,DTRTRAN={**CRTX**|name|NO}] [,DUMP={**YES**|NO}] [,DUMPDS={**AUTO**|A|B}] [,DUMPSW={**NO**|NEXT}] [,DURETRY={**30**|number-of-seconds|0}] [,ECDSASZE={**0K**|number}] [,EDSALIM={**20M**|number}] [,ENCRYPTION={WEAK|**NORMAL**|STRONG}] [,EODI={**E0**|xx}] [,ERDSASZE={**0K**|number}] [,ESDSASZE={**0K**|number}] [,ESMEXITS={**NOINSTLN**|INSTLN}] ***(SIT macro only)*** [,EUDSASZE={**0K**|number}] [,FCT={**NO**|xx|YES}] [,FEPI={**NO**|YES}] [,FLDSEP={**'_____'**|'xxxx'}] [,FLDSTRT={**'_'**|'x'}] [,FORCEQR={**NO**|YES}] [,FSSTAFF={YES|**NO**}] [,FTIMEOUT={**30**|nn}] [,GMTEXT={**'WELCOME TO CICS/ESA'**|'text'}] [,GMTRAN={**CSGM**|CESN|name}] [,GNTRAN={**NO**|transaction-id}] [,GRNAME=name] [,GRPLIST={**DFHLIST**|name|(name[,name2][,name3][,name4])}] [,GTFTR={**OFF**|ON}] [,HPO={**NO**|YES}] ***(SIT macro only)*** [,ICP=COLD] [,ICV={**1000**|number}] [,ICVR={**5000**|number}] [,ICVTSD={**500**|number}] [,INITPARM=(pgmname_1='parmstring_1' [, ... ,pgmname_n='parmstring_n'])] [,INTTR={**ON**|OFF}] [,IRCSTRT={**NO**|YES}] [,ISC={**NO**|YES}] [,JESDI={**30**|number}] [,KEYFILE={'*key-database-path-name*'} [,LGNMSG={**NO**|YES}]

Table 27. The DFHSIT macro parameters (continued)

		[,LLACOPY={**YES**|NO|NEWCOPY}] [,LPA={**NO**|YES}] [,MAXOPENTCBS={**5**|*number*}] [,MCT={**NO**|YES|xx}] [,MN={**OFF**|ON}] [,MNCONV={**NO**|YES}] [,MNEVE={**OFF**|ON}] [,MNEXC={**OFF**|ON}] [,MNFREQ={**0**|hhmmss}] [,MNPER={**OFF**|ON}] [,MNSUBSYS={**null**|xxxx}] [,MNSYNC={**NO**|YES}] [,MNTIME={**GMT**|LOCAL}] [,MQCONN={**NO**|YES}] [,MROBTCH={**1**|number}] [,MROFSE={**1**|number}] [,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}] [,PGAIEXIT={**DFHPGADX**|name}] [,PGAIPGM={**INACTIVE**|ACTIVE}] [,PGCHAIN=character(s)] [,PGCOPY=character(s)] [,PGPURGE=character(s)] [,PGRET=character(s)] [,PLTPI={**NO**|xx|YES}] [,PLTPISEC={**NONE**|CMDSEC|RESSEC|ALL}] [,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}] [,RAMAX={**256**|number}] [,RAPOOL={**2**|value1|(value1,value2)}] [,RDSASZE={**0K**|number}] [,RENTPGM={**PROTECT**|NOPROTECT}] [,RESP={**FME**|RRN}] [,RESSEC={**ASIS**|ALWAYS} [,RLS={**NO**|YES} [,RLSTOLSR={**NO**|YES}

Table 27. The DFHSIT macro parameters (continued)

		[,RMTRAN=({**GMTRAN-name**|name1} [,{**GMTRAN-name**|name2}])]
		[,RRMS={**NO**|YES}]
		[,RST={**NO**|xx|YES}]
		[,RUWAPOOL={**NO**|YES}]
		[,SDSASZE={**0K**|number}]
		[,SDTRAN={**CESD**|name_of_shutdown_tran|NO}]
		[,SEC={**YES**|NO}]
		[,SECPRFX={**NO**|YES}]
		[,SKRxxxx='page-retrieval-command']
		[,SNSCOPE={**NONE**|CICS|MVSIMAGE|SYSPLEX}]
		[,SPCTR={(**1,2**|1[,2][,3])|ALL|OFF}]
		[,SPOOL={**NO**|YES}]
		[,SRBSVC={**215**|number}]
		[,SRT={**1¢|**YES|xx|NO}]
		[,SSL={**NO**|YES|CLIENTAUTH}]
		[,START={**AUTO**|INITIAL|COLD|STANDBY}]
		[,STARTER={**NO**|YES}] ***(SIT macro only)***
		[,STATRCD={**OFF**|ON}]
		[,STGPROT={**NO**|YES}]
		[,STGRCVY={**NO**|YES}]
		[,STNTR={**1**|(1[,2][,3])|ALL|OFF}]
		[,SUBTSKS={**0**|1}]
		[,SUFFIX=xx] ***(SIT macro only)***
		[,SYDUMAX={**999**|number}]
		[,SYSIDNT={**CICS**|name}]
		[,SYSTR={**ON**|OFF}]
		[,TAKEOVR={**MANUAL**|AUTO|COMMAND}]
		[,TBEXITS=([name1][,name2][,name3] [,name4][,name5][,name6])]
		[,TCAM={**NO**|YES}]
		[,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}]
		[,TRTRANSZ={**16**|number}]
		[,TRTRANTY={**TRAN**|ALL}]
		[,TS=([COLD][,{0|**3**|value-1}][,{**3**|value-2}])]
		[,TST={**NO**|YES|xx}]
		[,UDSASZE={**0K**|number}]
		[,UOWNETQL=user_defined_value]
		[,USERTR={**ON**|OFF}]
		[,USRDELAY={**30**|number}]
		[,VTAM={**YES**|NO}]
		[,VTPREFIX={**|**|character}]
		[,WEBDELAY={**5**|time_out,**60**|keep_time}]

Table 27. The DFHSIT macro parameters (continued)

		[,WRKAREA={**512**|number}] [,XAPPC={**NO**|YES}] [,XCMD={**YES**|name|NO}] [,XDB2={**NO**|name}] [,XDCT={**YES**|name|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}] [,XRFSOFF={**NOFORCE**|FORCE}] [,XRFSTME={**5**|number}] [,XTRAN={**YES**|name|NO}] [,XTST={**YES**|name|NO}] [,XUSER={**YES**|NO}] You must terminate your macro parameters with the following END statement.
	END	DFHSITBA

DFHSIT, the default system initialization table

The macro source statements used to assemble the default system initialization table are given in Figure 48 on page 221. This default SIT is in CICSTS13.CICS.SDFHAUTH, and its source, named DFHSIT$$, is in CICSTS13.CICS.SDFHSAMP.

```
* $MOD(DFHSIT$$ COMP(STARTER) PROD(CICS/ESA):
*     5655-018                                                          *
*     COPYRIGHT = NONE                                                  *
*                                                                       *
*      PN= REASON REL YYMMDD HDXXIII : REMARKS                          *
*                                                                       *
* SIT parameters (in alphabetical order)                                *
*                                                                       *
SIT      TITLE 'DFHSIT - CICS DEFAULT SYSTEM INITIALIZATION TABLE'
         DFHSIT TYPE=CSECT,                                            *
               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                    *
               AUXTR=OFF,                 Auxiliary trace option               *
               AUXTRSW=NO,                Auxiliary trace autoswitch facility  *
               BMS=(FULL,,UNALIGN,DDS),   Basic Mapping Support options        *
               CICSSVC=216,               The CICS SVC number                  *
               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 *
               CSDACC=READWRITE,          CSD access                           *
               CSDBKUP=STATIC,            Backuptype of CSD (STATIC or DYNAMIC) *
```

Figure 48. DFHSIT, the pregenerated default system initialization table 1/9

```
CSDBUFND=,                  Number of data buffers for the CSD    *
CSDBUFNI=,                  Number of index buffers for the CSD   *
CSDDISP=,                   CSD Disposition for dynamic allocation*
CSDDSN=,                    CSD datasetname for dynamic allocation*
CSDFRLOG=NO,                Journal id. for CSD forward recovery  *
CSDINTEG=UNCOMMITTED,       Read integrity = uncommitted          *
CSDJID=NO,                  Journal id. for CSD auto. journaling  *
CSDLSRNO=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 *
DCT=NO,                     Dest. control table option/suffix     *
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  *
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=20M,                Upper limit of DSA above 16MB line    *
ENCRYPTION=NORMAL,          Level of encryption for SSL           *
EODI=E0,                    End-of-data indicator for seq. 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 builtin fn  *
FORCEQR=NO,                 Don't force QR for threadsafe progs.  *
FSSTAFF=NO,                 Function-shipped START affinity option*
FTIMEOUT=30,                File timeout 30 seconds               *
GMTEXT='WELCOME TO CICS/ESA', Good-morning message text           *
GMTRAN=CSGM,                Initial transaction                   *
GNTRAN=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 pgm. start option    *
ICV=1000,                   Region exit interval (milliseconds)   *
ICVR=5000,                  Runaway task interval (milliseconds)  *
ICVTSD=500,                 Terminal scan delay interval (  "  )  *
```

Figure 49. DFHSIT, the pregenerated default system initialization table 2/9

```
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  *
KEYFILE=,               Key database for SSL support          *
LGNMSG=NO,              Extract VTAM logon data               *
LLACOPY=YES,            Use MVS LLACOPY support               *
LPA=NO,                 Use-LPA option for CICS/user modules  *
MAXOPENTCBS=5,          Maximum number of open TCBs           *
MCT=NO,                 Monitoring cntl.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   *
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          *
PGAIEXIT=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=NO,               Program list table PI option/suffix   *
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,              Max. I/O area for RECEIVE ANY         *
```

Figure 50. DFHSIT, the pregenerated default system initialization table 3/9

```
RAPOOL=50,                Max. 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)    *
RUWAPOOL=NO,              Allocating storage pool for LE        *
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            *
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           *
```

Figure 51. DFHSIT, the pregenerated default system initialization table 4/9

```
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$,                 System recovery table option/suffix  *
SSL=NO,                 Secure sockets layer support         *
START=AUTO,             CICS system initialization option    *
STARTER=YES,            Starter ($ and #) suffixes option    *
STATRCD=OFF,            statistics recording status          *
STGPROT=NO,             Storage protection facility          *
STGRCVY=NO,             Storage recovery option              *
STNTR=1,                Level of standard tracing required   *
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                *
TCAM=NO,                TCAM option                          *
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*
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  *
XDB2=NO,                No RACF security profile for DB2     *
XDCT=YES,               DCT use default name for RACF check  *
XFCT=YES,               FCT use default name for RACF check  *
XJCT=YES,               JCT use default name for RACF check  *
```

Figure 52. DFHSIT, the pregenerated default system initialization table 5/9

```
         XLT=NO,                   Transaction list table option/suffix  *
         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*
         XRFSOFF=NOFORCE,          XRF -  Re-sign on after takeover      *
         XRFSTME=5,                XRF -  sign off timeout value         *
         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
END      DFHSITBA
```

Figure 53. DFHSIT, the pregenerated default system initialization table 6/9

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 CICSTS13.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, see “Chapter 2. Defining resources in CICS control tables” on page 7. For an explanation of the syntax notation used to describe CICS macros, see the manual.

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 54 on page 227.

For information about coding system initialization parameters in PARM, SYSIN, or at the console, see “Chapter 22. Processing system initialization parameters” on page 319.

```
CDSASZE={0K|number}
CHKSTRM={CURRENT|NONE}
CHKSTSK={CURRENT|NONE}
ECDSASZE={0K|number}
ERDSASZE={0K|number}
ESDSASZE={0K|number}
EUDSASZE={0K|number}
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 54. System initialization parameters you cannot code in the DFHSIT macro

Notes on CICS resource table and module keywords

Table 28 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 28. Summary of resources with a suffix, a dummy load module, or a COLD option

DFHSIT keyword 1	Default 2	Suffix 3	Dummy 4	COLD start 5
BMS	FULL	-	-	COLD
CLT	-	xx	-	-
DCT	NO	xx	-	-
DIP	NO	-	program	-
FCT	YES	xx	-	-
ICP	-	-	-	COLD
MCT	NO	xx	6	-
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 317.

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 295.

For more information about CICS table and program selection, see “Selecting versions of CICS programs and tables” on page 317.

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.

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.

AIEXIT={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 AIEXIT 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 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 via a CINIT, 0 means that the terminal entry is deleted as soon as the session is ended. For LU6.2 connections installed via 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 Performance Guide*.

AIRDELAY={700|hhmmss}
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

hhmmss
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 Performance 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. (The CICS region must support activity keypointing: that is, the CSKP transaction must be defined. For

more information about activity keypointing, see the *CICS Recovery and Restart Guide* and the *CICS Performance 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 103.

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 OS/390 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.

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.

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 264.

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 29 on page 235. 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 29. 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 *CICS Application Programming Guide*	All function of MINIMUM, **plus** 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 *CICS Application Programming Guide*	Same as STANDARD, **plus** terminal operator paging, cumulative mapping, page overflow, cumulative text processing, routing, message switching returning BMS-generated data stream to program before output.

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 CDSAZSE 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 Problem Determination 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 Problem Determination 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 the *CICS Transaction Server for OS/390 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 276.

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 Resource Definition Guide*.

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 30.

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 30. 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=YES)	User-key and ISOLATE(NO)	Task-lifetime storage of all **except** 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 CESF. 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 31 on page 239).

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 31.

Table 31. 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 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 CONFTXT parameter in the SIT, PARM, and SYSIN only.

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 150.

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 CSDLSRNO=NONE; if you have coded CSDLSRNO=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 CSDLSRNO=NONE; if you have coded CSDLSRNO=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 150.

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 150 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.

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 OS/390 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={2|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 143 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.

2 The minimum number of concurrent requests for the CSD is 2.

number
This number must be a multiple of 2, in the range 2 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 the *CICS System Programming Reference* manual.

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 either from CICS DB2 connection resource definitions installed from the CSD, or from an RCT specified by an INITPARM system initialization parameter.

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).

DCT=({YES|xx|NO})
specifies the destination control table suffix. (See page 227.) For information about defining this table, see the *CICS Resource Definition Guide*.

This parameter is effective only on a COLD start. CICS does not load a DCT on a warm or an emergency restart. All transient data destination definitions are recovered from the global catalog and from information provided by the recovery manager.

You can use a mixture of RDO and macro definitions for transient data, but you are recommended to use RDO. This is because there are new attributes available through RDO that are not available using the DFHDCT macro (for example, the INDOUBT attributes, WAIT and WAITACTION).

During a cold start, any macro entries defined in the macro load table are added to the transient data queue directory first. These are followed by RDO entries using GRPLIST. Any RDO entries being installed during a cold start that have the same name as entries already installed will be rejected. They can safely be installed once CICS is fully initialized.

You can no longer use the COLD option on the DCT system initialization parameter.

DFLTUSER={CICSUSER|userid}
specifies the RACF userid with the security attributes to be used for all terminal users who have not explicitly signed on (by the CESN transaction, the EXEC CICS SIGNON command, or by the preset security options of the TERMINAL resource definition).

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 227.)

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*.

DSHIPIDL={020000|hhmmss}
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.

hhmmss
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|hhmmss}
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 DELETSHIPPED or EXEC CICS SET DELETSHIPPED 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.

hhmmss
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*.

DSRTPGM
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.

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 175.

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 177.

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).

Restrictions You can specify the ECDSAZSE parameter in PARM, SYSIN, or CONSOLE only.

EDSALIM={20M|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.

20M The default EDSA limit is 20MB (20 971 520 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.

Note: For information about calculating the amount of storage to specify on the EDSALIM parameter, see the *CICS Performance Guide*.

ENCRYPTION={NORMAL|WEAK|STRONG}
Specifies the level of encryption you want to use for TCP/IP connections using the secure sockets layer. Possible values are:

NORMAL
Specifies a 56–bit encryption key, which is available worldwide except in France.

WEAK
Specifies a 40–bit encryption key, which is for use in France.

STRONG
Specifies a 128-bit encryption key. This is available when you have installed the North American encryption feature, which is available in the USA and Canada only.

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 ERDSAZSE 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 via 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.

EUDSASZE={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 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, and does not apply to Java programs that are run in a JVM.

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 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={'WELCOME TO CICS/ESA'|'text'}
specifies whether the default logon message text (WELCOME TO CICS/ESA) 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 OS/390 Release 3 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 OS/390 Release 3 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 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:

SIGNOFF
Effect

YES The terminal is signed off, but not logged off.

NO The terminal remains signed on and logged on.

LOGOFF
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.

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.

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*.

GTFTR={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 GTFTR=ON.

For information about GTF, see the *OS/390 MVS Diagnosis: Tools and Service Aids*manual, SY28-1085.

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 SVCPARM 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 OS/390 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 227 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.

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* manual.

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 324.

You can specify up to 255 pgmname='parmstring' sets.

Note: You can specify the INITPARM keyword and its parameters more than once, see 343.

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.

KEYFILE=*key-database-path-name*
Specifies the fully qualified HFS pathname of the key database created by the *gskkyman* utility program for this CICS region. When you specify this parameter, the CICS region userid must be authorised to read the specified HFS file.

LGDFINT={30|number
specifies the log defer interval 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.

30 This is the default.

number
number can be any value in the range 0 through 65535. You are recommended to allow LGDFINT to assume its default value, 30.

Note: The log defer interval can be modified dynamically by means of the CEMT SET SYSTEM command or EXEC CICS SET SYSTEM function, specifying the LOGDEFER option. However, it is not recommended that this value be modified in a production environment without a prior system evaluation and performance analysis of any changed value.

The CICS Log manager uses the log defer interval value when calculating how long to delay a forced journal write request before invoking the MVS system logger. This delay is required since MVS system logger IXGWRITE processing has a longer pathlength than the equivalent BSAM write macro call used in the old-style journal management of CICS/ESA R4.1.0 and earlier releases.

Performance evaluations of typical CICS transaction workloads have indicated that a value of 30ms gives a comparable internal transaction rate and CPU cost between the same workload on both CICS Transaction Server and CICS/ESA R4.1.0.

For CICS systems with many tasks issuing forced log write requests, these tasks will not be seen to be delayed for periods close to the log defer interval value, since on average a forced log write request will be issued while a log deferral is already being performed for another task.

Be aware that CICS performance can be adversely affected by a change to the log defer interval value. Too high a value will delay CICS transaction throughput due to the additional wait before invoking the MVS system logger. Although the range of possible values is from 0 to 65535ms, the default of 30ms should be considered to be the correct order of magnitude when setting the parameter in most cases.

A log defer interval value of less than 30ms will reduce the delay in CICS Log manager before invoking the IXGWRITE macro. This will improve the transaction response time, but will increase CPU cost for the system since CICS will buffer fewer journal requests into a given call to the MVS system logger, and so have to invoke the IXGWRITE macro more often.

Conversely, increasing the log defer interval value above 30 will increase the transaction response time since CICS will increase the delay period before invoking the IXGWRITE macro. However, more transactions will be able to write their own log data into the same log buffer before it is written to the MVS system logger and hence the total CPU cost of driving IXGWRITE calls will be reduced.

An example of a scenario where a reduction in the log defer interval might be beneficial to CICS transaction throughout would be where many forced log writes are being issued, and little concurrent task activity is occurring. Such tasks will spend considerable amounts of their elapsed time waiting for the log defer period to expire. In such a situation, there

is limited advantage in delaying a call to the MVS system logger to write out a log buffer, since few other log records will be added to the buffer during the delay period.

Note also that the log defer interval value is restored from the SIT after any CICS restart.

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* manual.

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.

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 CICSTS13.CICS.SDFHSAMP, in a member called DFHœUMOD. For details of the CICS system initialization parameter PRVMOD that you can use to override LPA=YES for selected modules, see page 281.

MAXOPENTCBS=5|*number*

Specifies the maximum number, in the range 1 to 999, of open TCBs that can exist concurrently in the CICS region. When specifying this value, take into account TCB storage requirements: TCBs use real storage, and virtual storage below 16MB.

The default value is **5** open TCBs.

CICS manages a pool of open TCBs up to the limit set by MAXOPENTCBS. At any one time, the pool can consist of some TCBs that are allocated, and others that are free. CICS attaches a new TCB only when there isn't a free TCB available in the pool. For example, if the maximum number of open TCBs is set at 100, the pool could consist of 50 open TCBs, not all of which are allocated.

If the demand for open TCBs is greater than the limit set by MAXOPENTCBS, tasks are queued waiting for a TCB.

MCT={NO|YES|xx}

specifies the monitoring control table suffix. (See page 227.) 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.

For information about coding the macros for this table, see the manual.

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 the *CICS System Programming Reference* manual.

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.

MNEVE={OFF|ON}
specifies whether SYSEVENT monitoring is to be made active during CICS initialization. The monitoring SYSEVENT status is recorded in the CICS global catalog for use during warm and emergency restarts.

OFF Set SYSEVENT monitoring to “not active”.

ON Set SYSEVENT monitoring to “active”.

For information about the implications for SYSEVENT recording in an MVS Workload Manager environment, see the *CICS Performance Guide*.

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*.

MNSUBSYS={null|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 MVS/ESA connection automatically during initialization.

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. 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={1|number}
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 re-attaching 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 is has been kept for a previous sequence of syncpoints).

It should be used with caution. For additional information, see the long running mirror sections of the *CICS Intercommunication Guide* and the *CICS Performance Guide*.

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.

1 All messages are to be printed.

0 Only critical errors or interactive messages are to be printed.

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 IIOP 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 32 on page 273.

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 32. 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 32. Languages and codes supported by CICS (continued)

NATLANG code	NLS code	Language
Note: **DBCS** denotes Double-Byte Character Set languages. The following language module suffixes are not supported by the message editing utility: • E - English master data sets. • K - Japanese data sets, where translation is performed by IBM. • C - Simplified Chinese data sets, where translation is performed by IBM. 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.

See “Chapter 8. Defining sequence numbering resources” on page 75 for information about named counter pools and the named counter options table, DFHNCOPT.

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 via 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 via PARM, SYSIN, or the console:

CSDACC
CSDBUFND
CSDBUFNI
CSDDISP
CSDDSN
CSDFRLOG
CSDINTEG
CSDJID

CSDLSRNO
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 "Classes of start and restart" on page 325.

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* manual.

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 via 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 227.) 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.

PGAIEXIT={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 the *CICS Supplied Transactions* manual.
5. CICS always processes a paging command entered by the operator before initiating a transaction in response to a macro request, that consists of either DFHBMS or DFHPC with the TRANSID operand coded.

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 227). 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 227).

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 ERRTERM|ERRTERM(name) parameter on EXEC CICS ROUTE commands to be operative. For programming information about notification of undelivered messages, see the *CICS Application Programming Reference* manual.

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 DFHVTAMP 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. This facility is not supported for TCAM devices.

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 the *CICS Supplied Transactions* manual.

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 OS/390 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 313.

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*.

RAMAX={256|value}
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*.

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).

RAPOOL={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).

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=({def.CSGM|name1}[,{def.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.

RRMS=NO|YES
specifies whether CICS is to register as a resource manager with Recoverable Resource Management Services (RRMS). If you use the External CICS Interface (EXCI) to communicate with this CICS system,and you wish to have DPL requests coordinated by RMS, you should specify RRMS=YES. Otherwise specify RRMS=NO.

RST={NO|xx|YES}
specifies a recoverable service table suffix. (See page 227.) 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* .

RUWAPOOL={NO|YES}
specifies the option for allocating a storage pool the first time an LE-conforming program runs in a task. .

NO CICS disables the option and provides no RUWA storage pool. Every EXEC CICS LINK to an LE-conforming application results in a GETMAIN for RUWA storage.

YES CICS creates a pool of storage the first time an LE-conforming program 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.

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).

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 33 on page 289.

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 RACF 1.9 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 the *CICS System Programming Reference* manual.

Table 33. 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.

SECPRFX={NO|YES}

specifies whether CICS is to prefix the resource names in any authorization requests to RACF with a prefix corresponding to the RACF userid for the CICS region. The prefix to be used (the userid for the CICS region) is obtained by the DFHIRP module.

NO CICS does not prefix the resource names in any authorization requests to RACF.

YES The RACF userid is used as the prefix for CICS resources defined to RACF. CICS prefixes the resource name in any authorization requests to RACF with a prefix corresponding to the RACF userid of the CICS region, obtaining the userid as follows:

- If you start CICS as a job, the prefix corresponds to the USER operand coded on the JOB statement of the CICS startup job stream.
- If you start CICS as a started task, the prefix corresponds to the RACF userid associated with the name of the start procedure in the RACF ICHRIN03 table.
- If you start a CICS job without an associated RACF userid, the prefix defaults to CICS.

For information about using the PREFIX option, see the *CICS RACF Security Guide*.

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 “DFHSIT, the default system initialization table” on page 220) is in CICSTS13.CICS.SDFHAUTH, and its source, named DFHSITœœ, is in CICSTS13.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 OS/390 Release 3.

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={(def.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. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels. For a list of all the available trace points and their level numbers, see the *CICS User's Handbook*. For information about the differences between special and standard CICS tracing, see the *CICS Problem Determination Guide*.

number
The level numbers for the level of special tracing you want for all CICS components. The 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={(def.1,2 |1[,2][,3])|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)). 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 the SPCTRxx keyword are shown in Table 34:

Table 34. CICS component names and abbreviations

Code	Component name	Code	Component name
AP	Application domain	BM	Basic mapping support
BF	Built-in function	BR	3270 Bridge

Table 34. CICS component names and abbreviations (continued)

Code	Component name	Code	Component name
CP	Common programming interface	DC	Dump compatibility layer
DD	Directory manager domain	DI	Batch data interchange
DM	Domain manager domain	DS	Dispatcher domain
DU	Dump domain	EI	Exec interface
EX	External CICS interface	FC	File control
GC	Global catalog domain	IC	Interval control
IS	Inter-system communication	KC	Task control
KE	Kernel	LC	Local catalog domain
LD	Loader domain	LG	Log manager domain
LM	Lock domain	ME	Message domain
MN	Monitoring domain	NQ	Enqueue domain
PA	Parameter domain	PC	Program control
PG	Program manager domain	RI	Resource manager interface (RMI)
RM	Recovery manager domain	SC	Storage control
SM	Storage domain	ST	Statistics domain
SZ	Front end programming interface	TC	Terminal control
TD	Transient data	TI	Timer domain
TR	Trace domain	TS	Temporary storage domain
UE	User exit interface	US	User domain
WB	WEB interface	XM	Transaction manager domain
XS	Security manager domain		

Note: The component codes BF, BM, CP, DC, DI, 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 will be produced with a point ID of AP nnnn.

For details of using trace, see the *CICS Problem Determination Guide*.

number
The level numbers for the level of special tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

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.

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.

Note: If you use the CICS spool interface, this makes use of the MVS exit IDFDOIXT, which is provided in the SYS1.LINKLIB library. If you have a

high volume spool output, you should install the IDFDOIXT exit in a library in the CICS STEPLIB concatenation, and consider having a PLT startup program MVS load the exit during CICS initialization. This will help optimize the performance of the CICS spool interface.

For further information about the MVS exit IEFDOIXT, see the *OS/390 MVS Installation Exits*, SC28-1753.

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 OS/390 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 227.) 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.

SSLDELAY={600|number}
specifies the SSL time delay in seconds in the range 0 thru 86400.

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 232) 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 .

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 28 on page 227.

Restrictions You can specify START=(option,ALL) in PARM, SYSIN, or CONSOLE only.

For more information about the types of CICS startup, see “Classes of start and restart” on page 325.

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.

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, Unsolicited and Requested statistics are written to SMF regardless of the STATRCD setting.

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 the *CICS Supplied Transactions* manual. 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 37 on page 356 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 37 on page 356 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 the STGRCVY(YES) system initialization parameter, CICS resets the check zones correctly and the task continues running.

STNTR={1|(1[,2][,3])|ALL|OFF}
specifies the level of standard tracing required for CICS as a whole. You can select up to three levels of tracing, but some CICS components do not have trace points at all these levels.

Note: Before globally activating tracing levels 3 and ALL for the storage manager (SM) component, read the warning 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={(1,21[,2][,3])|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 34 on page 291.

number
The level number(s) for the level of standard tracing you want for the CICS component indicated by xx. The options are: 1, (1,2), or (1,2,3).

ALL You want all the available levels of standard tracing switched on for the specified component.

OFF Switches off all levels of standard CICS tracing for the CICS component indicated by xx.

Note: If you select tracing levels 3 or ALL for the storage manager (SM) component, or the temporary storage domain (TS), the performance of your CICS system will be degraded. This is because options 3 and ALL switch on levels of trace that are also used for field engineering purposes. See the *CICS Problem Determination Guide* for information about the effects of trace levels 3 and ALL.

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 may be taken per dump table entry. If this number is exceeded, subsequent system 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.

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}
specifies whether TCAM support is to be included.

NO TCAM support is not to be included.

YES TCAM support is to be included.

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 and IIOP services can process work.

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 parameter only applies to VTAM terminals (including LU Type 6.2

single-session APPC terminals), not VTAM intersystem connections (LU Type 6.1 and LU Type 6.2 parallel connections). 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.

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.

FORCE
CICS terminal control attempts to forceclose the CICS VTAM ACB if there are any hung terminals. All CICS VTAM terminals and sessions are released and CICS normal shutdown continues.

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)).

TCT={NO|xx|YES}
specifies which terminal control table, if any, is to be loaded. (See page 227.) 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 318. (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 354 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 153.

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 when OFFSITE=YES is specified.

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 replace a corrupt TD intrapartition data set.

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.

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 as in earlier releases. 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).

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.

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 264.

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 227.)

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 UDSAZSE 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.

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 OS/390-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. Only the BINDPASSWORD (defined to CICS for an APPC connection) is checked.

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. If it does not exist, only the BINDPASSWORD (defined to CICS for the connection) is checked.

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 C*name* and the grouping class name is V*name*.

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 X***aaa*** 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 CICSDCT 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 D*name* and the grouping class name is E*name*.

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.

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 F*name* and the grouping class name is H*name*.

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. (See page 227.) 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 M*name* and the grouping class name is N*name*.

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 282.

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 P*name* and the grouping class name is Q*name*.

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}

specifies whether all users signed-on to the active CICS region are to remain signed-on following a takeover. This parameter is only applicable if you also code XRF=YES as a system initialization parameter.

NOFORCE

Allow CICS to determine sign-off according to the option set in either of:

- The CICS segment of the RACF database
- The TYPETERM definition for the user's terminal

Note: For a terminal to remain signed-on after an XRF takeover, NOFORCE must be specified in the SIT, RACF database, and the terminal's TYPETERM definition.

FORCE

All terminal users are to be signed off in the event of a takeover by an alternate CICS region, regardless of individual options set in the RACF database or in the terminals' TYPETERM definitions.

For information about the XRFSOFF option on the CEDA DEFINE TYPETERM command, see the *CICS Resource Definition Guide*.

XRFSTME={5|decimal-value}

specifies, in minutes, a time-out delay interval for users who are still signed-on when an XRF takeover occurs.

If you have specified NOFORCE in the RACF database, and in the terminals' TYPETERM definitions, and the takeover takes longer than the time specified in the XRFSTME, all users who are still signed-on after takeover are signed off.

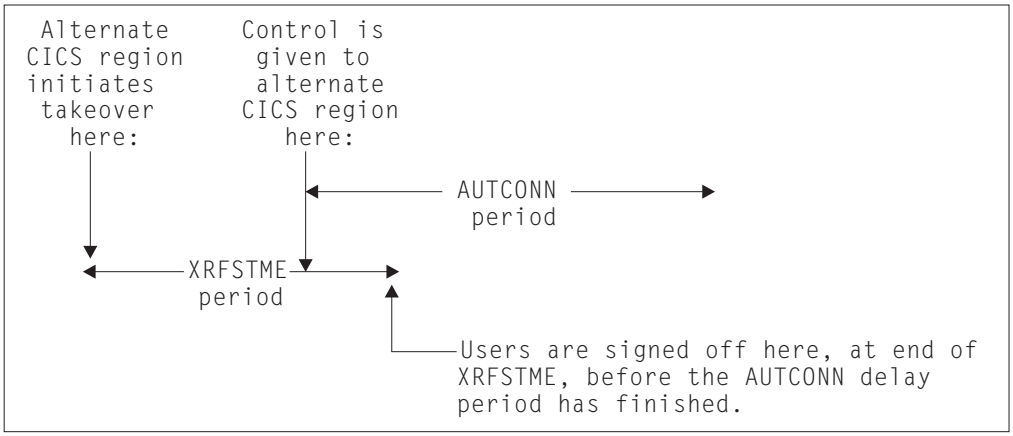
5 Five minutes is the default value in the DFHSIT macro.

decimal-value

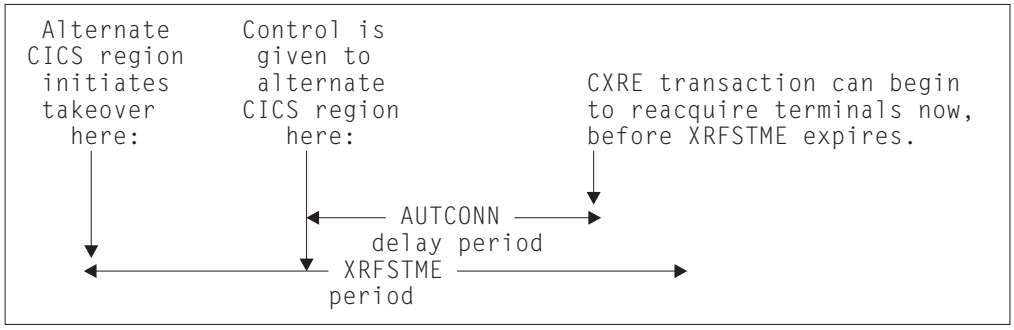
A value in the range 0 through 60 for the number of minutes CICS permits users to remain signed on during the takeover period. The takeover period is the time from when the takeover is initiated to the time at which CICS is ready to process user transactions. If the takeover takes longer than the specified period, all users signed-on at the time the takeover was initiated are signed-off.

A value of 0 specifies that there is no time-out delay, and terminals are signed off as soon as takeover commences, which means that XRFSTME=0 has the same effect as coding XRFSOFF=FORCE.

For non-XRF-capable terminals, take into account any AUTCONN delay period when setting the value for XRFSTME. (See the description of the AUTCONN parameter on page 233.) You may need to increase the XRFSTME value to allow for the delay to the start of the CXRE transaction imposed by the AUTCONN parameter; otherwise, terminals may be signed-off too early. For example:



You can avoid this situation by extending the XRFSTME period to exceed the AUTCONN period. For example:



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 and specified the RESSEC(YES) option on the resource definitions.

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 T*name* and the corresponding grouping class name is G*name*.

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.

Note: The checking is performed only if you have specified YES for the SEC system initialization parameter.

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.

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” for details).

You can use these methods **only** for the programs referred to in this section and in “Excluding unwanted programs”, 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 *programname*=NO as a system initialization parameter, (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 program's control table

Not all of the CICS programs have a programname parameter in the SIT. An alternative method is to code NO on the system initialization parameter 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 pregenerated dummy table of this name is supplied in CICSTS13.CICS.SDFHLOAD, and the source statements of DFHTCTDY are supplied in CICSTS13.CICS.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, TCAM support, or the extended recovery facility (XRF), in this way.

Chapter 22. 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

Methods of 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, parameter manager domain 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 page 321)
3. The console (but only if CONSOLE is coded in either the PARM parameter or in the SYSIN data set; see page 321)

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$.

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 via 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 275.

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.

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=DFHSIP 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 321.)

Examples:

```
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,SYSIN,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,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=DFHSIP 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=DFHSIP,PARM='SIT6$,CONSOLE,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='CONSOLE,SYSIN,.END'
//stepname   EXEC  PGM=DFHSIP,PARM='SIT=6$,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, depending on its context:

Context	Explanation
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=DFHSIP,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 simple 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.

Classes of 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 at the beginning of this chapter.

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".

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 "The local catalog" on page 165. 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 are the system initialization parameters DCT, 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 35 shows how the effect of the START parameter depends on the state of the CICS global catalog and system log.

Table 35. 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 [7] 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 [7] are initialized.
COLD	Contains recovery manager records.	Not defined or dummy or empty.	Message DFHRM0401 is issued. Startup fails.

7. If the system log is defined as a dummy, it is ignored.

Table 35. 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
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 [7] are initialized.
AUTO	Contains a recovery manager control record that does *not* 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 [7] are initialized.
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 36 on page 331 shows the effect of different types of CICS startup on the CICS trace, monitoring, statistics, and dump domains.

Table 36. 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.
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, but the dump status is 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=xxxxxxxx, 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* manual.

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=xxxxxxxx, ACB CODE=yy.
DFHSI1590 'applid'  XRF alternate cannot proceed without VTAM.
```

CICS abends with a dump (abend code 1590).

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 the *CICS Supplied Transactions* manual.

Chapter 23. Defining the CICS JVM execution environment variables

This chapter describes how to define the default environment variables that control the initialization of the CICS Java® Virtual Machine.

CICS supplies a partitioned dataset SDFHENV containing default values of the environment variables. A DD statement for this dataset must be included in your CICS startup jobstream if you intend to run CICS Java applications that execute in the JVM. Such programs have JVM(YES) in their PROGRAM resource definition. The following DD statement is required:

```
//DFHJVM   DD DSN=CICSTS13.CICS.SDFHENV(DFHJVM),DISP=SHR
```

You can edit this file with TSO to change the default values The user replaceable module DFHJVMAT can also be called at JVM initalization to examine and reset the values. See the *CICS Customization Guide* for a description of DFHJVMAT.

JVM environment variables

The following contents of SDFHENV are shipped in CICS Transaction Server for OS/390 Release 3:

```
CHECKSOURCE=NO
CICS_HOME=.
CLASSPATH=/usr/lpp/cicsts///cicsts13///classes/dfjwrap.jar
   :/usr/lpp/cicsts///cicsts13///classes/dfjcics.jar
   :/usr/lpp/cicsts///cicsts13///classes/dfjcorb.jar
   :/usr/lpp/java/J1.1/lib/classes.zip
DISABLEASYNCGC=NO
ENABLECLASSGC=YES
ENABLEVERBOSEGC=NO
INVOKE_DFHJVMAT=NO
JAVASTACKSIZE=409600
JAVA_COMPILER=OFF
JAVA_HOME=/usr/lpp/java/J1.1
LIBPATH=/usr/lpp/cicsts///cicsts13///lib
   :/usr/lpp/java/J1.1/lib/mvs/native_threads
MAXHEAPSIZE=8000000
MINHEAPSIZE=1000000
NATIVESTACKSIZE=262144
STDERR=dfhjvmerr
STDIN=dfhjvmin
STDOUT=dfhjvmout
VERBOSE=NO
VERIFYMODE=NO
```

Where:

CHECKSOURCE
Tells the JVM to check the source file and **.class** file. If the **.class** file is out of date, the JVM recompiles the source.

CICS_HOME
specifies the HFS directory that is used by the CICS JVM interface when creating **stdin**, **stdout** and **stderr** files. A period (.) is defined in SDFHENV, which means that the current directory will be used. If CICS_HOME is not set at all, then /tmp will be used as the directory.

CLASSPATH
Sets the directory path to be searched by the JVM for .class files. The default shipped in SDFHENV includes dfjcics.jar and dfjwrap.jar (required to support JCICS).

DISABLEASYNCGC
Indicates whether asynchronous garbage collection should be disabled.

ENABLECLASSGC
Indicates whether the JVM should perform garbage collection on loaded classes that are not being used.

ENABLEVERBOSEGC
Indicates whether the JVM should issue a message when the garbage collector frees memory.

INVOKE_DFHJVMAT
Specifies whether the user replaceable module DFHJVMAT should be invoked before executing the JVM.

JAVASTACKSIZE
Sets the size of each thread's Java code stack, in bytes. A default value of 409600 bytes (400K) is set in SDFHENV. This is the recommended default for the MVS JVM.

JAVA_COMPILER
Specifies whether the Java just-in-time (JIT) compiler should be invoked by the JVM.

JAVA_HOME
Defines the installation directory prefix of the JDK.

LIBPATH
Sets the directory path to be searched by the JVM for native C `dll` files. The default directory path set in SDFHENV includes the path to the native C dll files required to support JCICS.

MAXHEAPSIZE
Sets the maximum heap size for the JVM, in bytes. A default value of 8000000 bytes (8M) is set in SDFHENV. This is the recommended default for the MVS JVM. This is above-the-line storage.

MINHEAPSIZE
Sets the minimum heap size for the JVM, in bytes. A default value of 1000000 bytes (1M) is set in SDFHENV. This is the recommended default for the MVS JVM. This is above-the-line storage.

NATIVESTACKSIZE
Sets the size of each thread's stack, in bytes. A default value of 262144 bytes (256K) is set in SDFHENV. This is the recommended default for the MVS JVM.

Note: The initial process thread (IPT) stack is governed by the STACK run-time option on the DFHCJVM C program that invokes the JVM. This program sets the following value:

```
#pragma runopts(STACK(64K,16K,ANYWHERE,KEEP))
```

STDERR
specifies the name of the HFS file to be used for **stderr**. The default shipped in SDFHENV is dfhjvmerr. The file will be created if it does not exist. If the file already exists, output is appended at the end of the file. On completion of the JVM program, if the stderr file is empty, it is deleted.

STDIN
specifies the name of the HFS file to be used for **stdin**. The default shipped in SDFHENV is dfhjvmin. The will be created if it does not exist.

STDOUT
specifies the name of the HFS file to be used for **stdout**. The default shipped in SDFHENV is dfhjvmout. The file will be created if it does not exist. If the file already exists, output is appended at the end of the file. On completion of the JVM program, if the stdout file is empty, it is deleted.

VERBOSE
Indicates whether the JVM should issue a message each time it loads a class.

VERIFYMODE
Indicates whether the bytecode verifier should be run on all classes that are loaded.

Chapter 24. 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 339. “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. For more information about storage protection, see “Storage protection” on page 353. For programming information about writing PLT programs, see the *CICS Customization Guide*.

If you are running CICS with DB2, you can specify the resource control table suffix and DB2 subsystem ID to be used at startup by the INITPARM system initialization parameter, as follows:

```
INITPARM=(DFHD2INI='xx,yyyy')
```

where xx is the 2-character resource control table suffix and yyyy is the 4-character DB2 subsystem ID. Both values must conform to MVS JCL rules about special characters. If you specify a DB2 subsystem ID, it is used at PLT startup (with the resource control table suffix specified).

For information about setting up the RCT, see the *CICS DB2 Guide*.

Sample startup job stream

You can use the sample job control statements shown in Figure 55 on the following pages to initialize a CICS region. The sample job stream shown in Figure 55 specifies START=AUTO and XRF=YES to start an active CICS region. The notes that follow Figure 55 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 91.

JCL similar to that in Figure 55 on page 339 is supplied as a sample startup procedure, DFHSTART, in the CICSTS13.CICS.SDFHINST library. You can use the DFHSTART procedure to start the CIC regions, or as a basis for your own startup procedures. For information about the DFHSTART procedure, see the *CICS Transaction Server for OS/390 Installation Guide*.

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,
*
CICSDFLTUSER=CICSUSER,   The default userid
MXT=30,              Maximum number of user tasks is 30
INITPARM=(DFHDBCON='01',DFHD2INI=('01,MYDB')),
                     Pass DFSPZP01 suffix to DBCTL connect program
                     Use RCT DFHRCT01 with DB2 subsystem MYDB
ISC=YES,             Include intersystem communication program
IRCSTRT=YES,         Start interregion communication
.END
/*
//*
```

Figure 55. CICS startup job stream 1/3

```
//*  6    The STEPLIB library
//STEPLIB  DD  DSN=CICSTS13.CICS.SDFHAUTH,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//*
//*  7    The CICS library (DFHRPL) concatenation
//DFHRPL   DD DSN=CICSTS13.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 LE run-time data sets
//         DD  DSN=CEE.SCEECICS,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//*  7a   The DB2 load library
//         DD  DSN=SYS2.DB2.SDSNLOAD,DISP=SHR
//*  8    Auxiliary temporary storage data sets
//DFHTEMP  DD DSN=CICSTS13.CICS.CNTL.CICSHTH1.DFHTEMP,DISP=SHR
//*
//*  9    Intrapartition data sets
//DFHINTRA DD DSN=CICSTS13.CICS.CNTL.CICSHTH1.DFHINTRA,DISP=SHR
//*
//*  10   The auxiliary trace data sets
//DFHAUXT  DD DSN=CICSTS13.CICS.CICSHTH1.DFHAUXT,DISP=SHR
//DFHBUXT  DD DSN=CICSTS13.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=CICSTS13.CICS.CICSHTH1.DFHLCD,DISP=SHR
//*  13   The CICS global catalog data set
//DFHGCD   DD DSN=CICSTS13.CICS.CICSHTH1.DFHGCD,DISP=SHR,
//            AMP=('BUFND=33,BUFNI=32,BUFSP=303104')
//*
//*  14   The CAVM data sets - XRF
//DFHXRMSG DD DSN=CICSTS13.CICS.CNTL.CICSHTH1.DFHXRMSG,DISP=SHR
//DFHXRCTL DD DSN=CICSTS13.CICS.CNTL.CICSHTH1.DFHXRCTL,DISP=SHR
//*
```

Figure 56. CICS startup job stream 2/3

```
//* 15    The transient data destination data set (CXRF)
//DFHCXRF  DD SYSOUT=A
//*
//* 16    The CICS transaction dump data sets
//DFHDMPA  DD DSN=CICSTS13.CICS.CICSHTH1.DFHDMPA,DISP=SHR
//DFHDMPB  DD DSN=CICSTS13.CICS.CICSHTH1.DFHDMPB,DISP=SHR
//*
//* 17    MVS system dump data sets
//SYSABEND DD SYSOUT=*
//SYSMDUMP DD  DSN=SYS1.SYSMDP00,VOL=SER=volid,SPACE=(CYL,(1,1)),
//             DISP=OLD,UNIT=3380
//* SYSUDUMP DD SYSOUT=*
//*
//* 18    The CICS system definition data set
//DFHCSD   DD DSN=CICSTS13.CICS.DFHCSD,DISP=SHR
//*
//* 19    The CICS BTS local request queue data set
//DFHLRQ   DD DSN=CICSTS13.CICS.DFHLRQ,DISP=SHR
//*
//* 20    The JVM environment variables data set
//DFHJVM   DD DSN=CICSTS13.CICS.SDFHENV(DFHJVM),DISP=SHR
//*
//* 21    The JVM dummy file
//DFHCJVM  DD DUMMY
//*
//* 22    The CMAC data file
//DFHCMACD DD DSN=CICSTS13.CICS.DFHCMACD,DISP=SHR
//*
//* 23    The CDBM group command file
//DFHDBFK  DD DSN=CICSTS13.CICS.DFHDBFK,DISP=SHR
//*
//* 24    FILEA & other permanently allocated data sets
//* (The FILEA DD statement below overrides the CSD definition in
//*  group DFHMROFD)
//FILEA    DD DISP=SHR,
// DSN=CICSTS13.CICS.CICSHTH1.FILEA
//*
//APPLICA  DD  DSN=CICSTS13.CICS.CICSHTH1.APPLICA,DISP=SHR
//APPLICB  DD  DSN=CICSTS13.CICS.CICSHTH1.APPLICB,DISP=SHR
//*
//*
//* 25
//PRINTER  DD  SYSOUT=A,DCB=BLKSIZE=125,OUTLIM=0
//CARDIN   DD  *
   :
\user transactions input from sequential terminal \
   :
\CESF GOODNIGHT\
/*
```

Figure 57. 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 351.

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 run-time 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 285.

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 OS/390 Program Directory*that you have the required hardware and software. See page 296 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=(DSN2STRT= '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 VS COBOL II 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 SVCPARM statement. For more information about the CICSSVC parameter, see page 236.

For guidance information about installing the CICS SVC in the LPA, and defining it to MVS, see the *CICS Transaction Server for OS/390 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 272.

INITPARM
Passes parameters to programs. In this example the DBCTL suffix (01) of DFSPZPxx is passed to DFHDBCON, and the resource table DFHRCT01 is used with the DB2 subsystem MYDB.

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 CICSTS13.CICS.SDFHAUTH.

Placing user-written modules into an APF-authorized library may violate your security and integrity rules. The CICSTS13.CICS.SDFHAUTH library should not be included in the MVS link list, so that you can protect the library 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 CICSTS13.CICS.SDFHAUTH 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 CICSTS13.CICS.SDFHAUTH.

You also need the Language Environment run-time library, CEE.SCEERUN, in the STEPLIB concatenation (or the MVS linklist) to run COBOL, PL/I, C and C++, and JVM programs under LE. Like SDFHAUTH, SCEERUN must be an APF-authorized library.

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 CICSTS13.CICS.SDFHLOAD, which must be included in the CICS DFHRPL library concatenation. The CICSTS13.CICS.SDFHLOAD library contains only programs that run in problem state and should not be authorized.

LE run-time library requirements

You are recommended to use the LE run-time libraries to support all your program languages, including the older versions, such as VS COBOL II and PL/I Versions 1 and 2. To do this, add the Language Environment run-time library SCEERUN to the DFHRPL DD concatenation. If you are running COBOL programs, also add Language Environment run-time library SCEECICS to DFHRPL. The SCEECICS library must be concatenated before the SCEERUN library. Remove any libraries that contain run-time routines from earlier versions of COBOL, PL/I, and C/C++ from the DFHRPL DD concatenation.

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 10. Defining the temporary storage data set” on page 107.

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 26. Starting up temporary storage servers” on page 367

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 the intrapartition data set” on page 115.

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 15. Defining and using auxiliary trace data sets” on page 171.

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 38 on page 173, you can define them to CICS in the startup job stream using the following DD statements:

```
//DFHAUXT  DD  DSN=CICSTS13.CICS.applid.DFHAUXT,DCB=BUFNO=n,DISP=SHR
//DFHBUXT  DD  DSN=CICSTS13.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 14. Defining and using catalog data sets” on page 159.

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 14. Defining and using catalog data sets” on page 159.

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 35 and the associated notes given under 4 on page 160. 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 17. Defining the CICS availability manager data sets” on page 181.

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 118.

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 16. Defining dump data sets” on page 175.

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 55 on page 339), 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.

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 13. Defining the CICS system definition data set" on page 135.

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 and availability of the CSD in non-RLS mode” on page 140.

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 CICS JVM environment variables data set

The JVM environment variables are specified in a PDS member referenced by the DFHJVM DD statement. This member contains the information needed by CICS to initialize a JVM to execute a JVM program. For information about DFHJVM and its contents, see “Chapter 23. Defining the CICS JVM execution environment variables” on page 333

21 The CICS JVM dummy

CICS requires this dummy DD statement when creating a JVM to execute a JVM program.

22 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 20. Defining the CMAC messages data set” on page 211 describes DFHCMACD in greater detail.

23 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.

on page 207 describes DFHDBFK in greater detail.

24 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 18. Defining user files” on page 189. 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 CICSTS13.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*.

25 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. For more information about how to restart a device that has been closed in a previous run of CICS by means of a CESF GOODNIGHT transaction, see page 67.

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*.

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 ERDSA 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$$, a default system initialization table. This default table is supplied in the CICSTS13.CICS.SDFHSAMP library in source form, and you can modify this to suit your own requirements. When assembled and link-edited, DFHSIT$$ becomes the unsuffixed DFHSIT, which is supplied in pregenerated form in CICSTS13.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 245 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 TCAM and 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 302 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 296 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 285 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:

CDSA
The CICS DSA, allocated below the 16MB boundary, always from CICS-key storage.

RDSA
The read-only DSA, allocated below the 16MB boundary from either read-only storage or CICS-key storage depending on the RENTPGM parameter.

SDSA
The shared DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.

UDSA
The user DSA, allocated below the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.

ECDSA
The extended CICS-key DSA, allocated above the 16MB boundary, always from CICS-key storage.

ERDSA
The extended read-only DSA, allocated above the 16MB boundary, either from read-only storage or CICS-key storage, depending on the RENTPGM parameter.

ESDSA
The extended shared DSA, allocated above the 16MB boundary from either user-key or CICS-key storage depending on the STGPROT parameter.

EUDSA
The extended user DSA, allocated above the 16MB boundary, from either user-key or CICS-key storage depending on the STGPROT parameter.

Table 37 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 37. 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[1]	N/A[1]
RDSA	N/A[2]	N/A[2]	Read-only key-0	CICS key
SDSA	CICS key	User key	N/A[1]	N/A[1]
UDSA	CICS key	User key	N/A[1]	N/A[1]
ECDSA	CICS key	CICS key	N/A[1]	N/A[1]
ESDSA	CICS key	User key	N/A[1]	N/A[1]

Table 37. Controlling the storage key for the dynamic storage areas (continued)

Dynamic storage area	STGPROT= NO	STGPROT= YES	RENTPGM= PROTECT	RENTPGM= NOPROTECT
ERDSA	N/A[2]	N/A[2]	Read-only key-0	CICS key
EUDSA	CICS key	User key	N/A[1]	N/A[1]
Note: 1. Not applicable. The RENTPGM option has no effect on these DSAs, for which the storage key is determined only by the STGPROT parameter. **Note:** 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.				

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 248 and 253 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 38:

Table 38. 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.

The sample program consists of the following resources:

DFH0STAT
Statistics program, VS COBOL II release 2 or later.

DFH$STAS
Assembler language program called by DFH0STAT.

DFH$STCN
Assembler language program called by DFH0STAT.

DFH$STTB
Assembler language table loaded by DFH0STAT.

DFH0STM
Mapset used by STAT transaction.

DFH0STS
Mapset used by the STAT transaction.

STAT Transaction used to invoke DFH0STAT.

The sample program can be invoked as follows:

- As a PLTPI program (after DFHDELIM)
- As a PLTSD program (before DFHDELIM)
- As a conversational transaction from a terminal
- From a console

- As a started transaction using the EXEC CICS START command from a user-written application program
- By a distributed program link request from a user-written application program

To enable you to use the sample program, you must:

1. Assemble and link-edit the BMS mapsets DFH0STM and DFH0STS. Include the physical mapset in a library that is in the DFHRPL concatenation. You can either include the symbolic map set in a user copy library or insert it directly into the source for DFH0STAT.

   For more information about installing and using map sets, see “Chapter 3. Installing map sets and partition sets” on page 13 and “Using BMS map sets in application programs” on page 34.
2. Translate the DFH0STAT program source code, turning CICS commands into code understood by the compiler. The program source code is provided in CICSTS13.CICS.SDFHSAMP.
3. Compile the translator output for DFH0STAT to produce object code.
4. Link-edit the object module to produce a load module, which you store in an application load library that is concatenated to the DFHRPL DD statement of the CICS startup job stream.
5. Create resource definition entries, in the CSD, for the programs, the mapset, and the STAT transaction. The CICS-supplied RDO group, DFH$STAT, contains all the necessary resource definitions for this sample.
6. Define the system initialization parameter SPOOL=YES. This specifies that you need support for the system spooling interface.

For more information about installing programs, see “Chapter 4. Installing application programs” on page 23.

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=subystemname
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, CICSTS13.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, CICSTS13.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 OS/390 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*.

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 25. Defining and starting AXM system services” on page 365 describes how to define and start AXM system services
- “Chapter 26. Starting up temporary storage servers” on page 367 describes how to start a CICS temporary storage data sharing server
- “Chapter 27. Starting a coupling facility data table server” on page 381 describes how to start a CICS coupling facility data table server
- “Chapter 28. Starting a named counter server” on page 403 describes how to start a CICS named counter server.

Chapter 25. Defining and starting AXM system services

This chapter describes how to define and start AXM system services, which are a 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 IEFSSN*xx* 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 26. Starting up temporary storage servers

This chapter describes how to start up temporary storage servers, and provides information about the following:

Note: See "Defining temporary storage pools for temporary storage data sharing" on page 110 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, XTSEREQ, to modify the SYSID on a TS request so that it references a TS data sharing pool.

Figure 58 on page 368 illustrates a parallel sysplex with three CICS AORs linked to the temporary storage server address space(s).

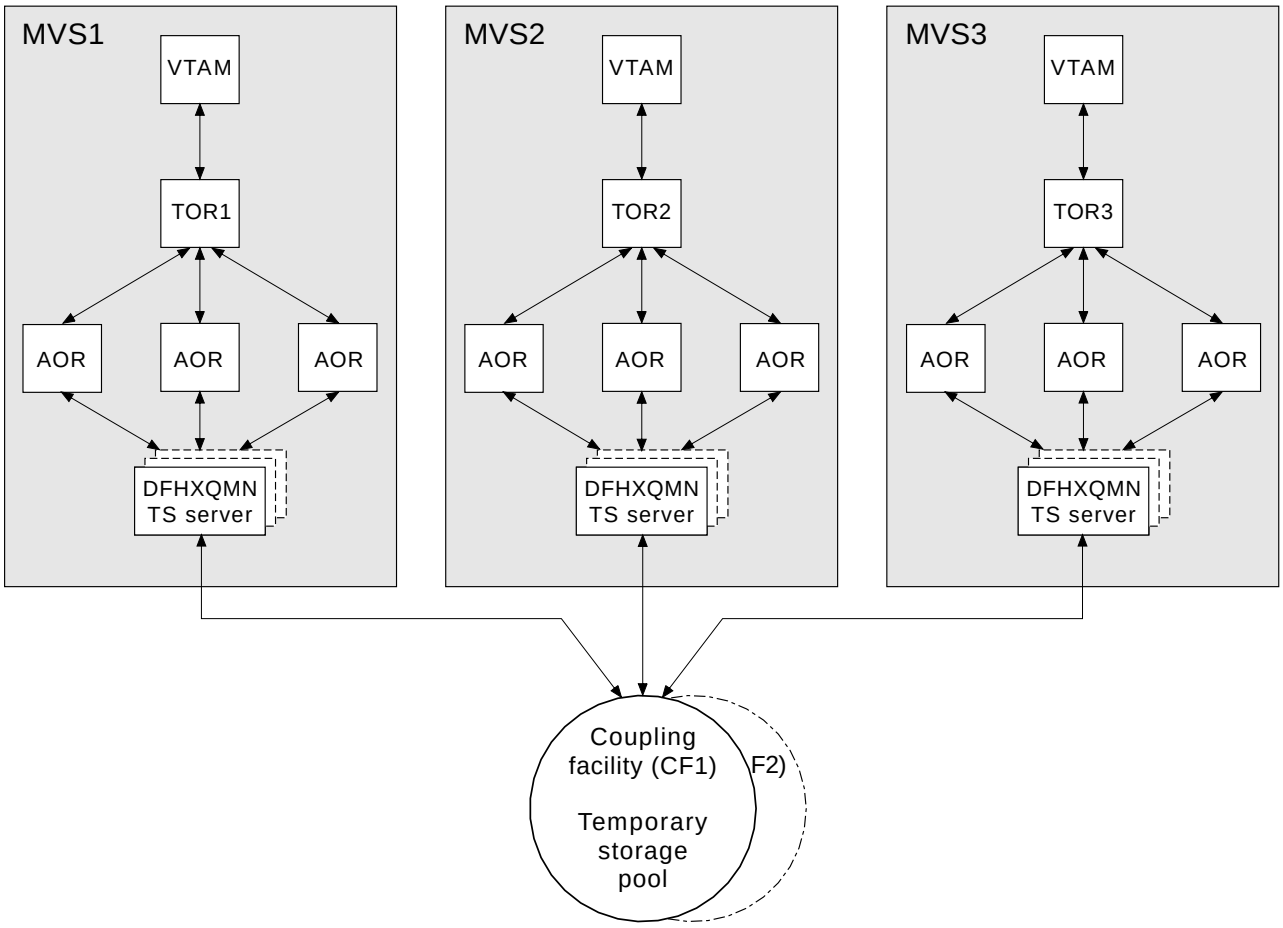


Figure 58. Conceptual view of a Parallel Sysplex with TS data sharing.

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 59 shows an example of the JCL you might use to start a TS server.

```
//PRODTSQ1 JOB  ...
//TSSERVER EXEC PGM=DFHXQMN,REGION=64M Start CICS 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)
/*
```

Figure 59. 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 377.

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**.

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 *n*K, or megabytes in the form *n*M, or gigabytes in the form *n*G.

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.

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 110.

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 **ELEMRATIO**.

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|hhmm}
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” on page 375.

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
terminate the server immediately.

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:

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**.

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. The server does not support automatic structure REBUILD, but the unload and reload process is more flexible. The reloaded queue pool does not need the same name as the original pool.

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 60 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 60. Example of UNLOAD JCL

The RELOAD function requires a DD statement for file name DFHXQRL 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 61 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 61. Example of RELOAD JCL

TS server message definitions

The two categories of messages produced by the TS server are as follows:

- DFHXQxxxx messages issued by TS queue server modules (usually in cross memory mode).
- AXMxx messages issued by the authorized cross-memory (AXM) server environment. These messages are not issued by a CICS region, and do not use the CICS message domain.

For full details about messages produced by the TS server, see the *CICS Messages and Codes* manual.

Chapter 27. Starting 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 382
- "Controlling coupling facility data table server regions" on page 394
- "Deleting or emptying coupling facility data table pools" on page 399
- "Unloading and reloading coupling facility data table pools" on page 400.

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 205 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 62 on page 382 illustrates a Parallel Sysplex with three CICS AORs linked to the coupling facility data table servers.

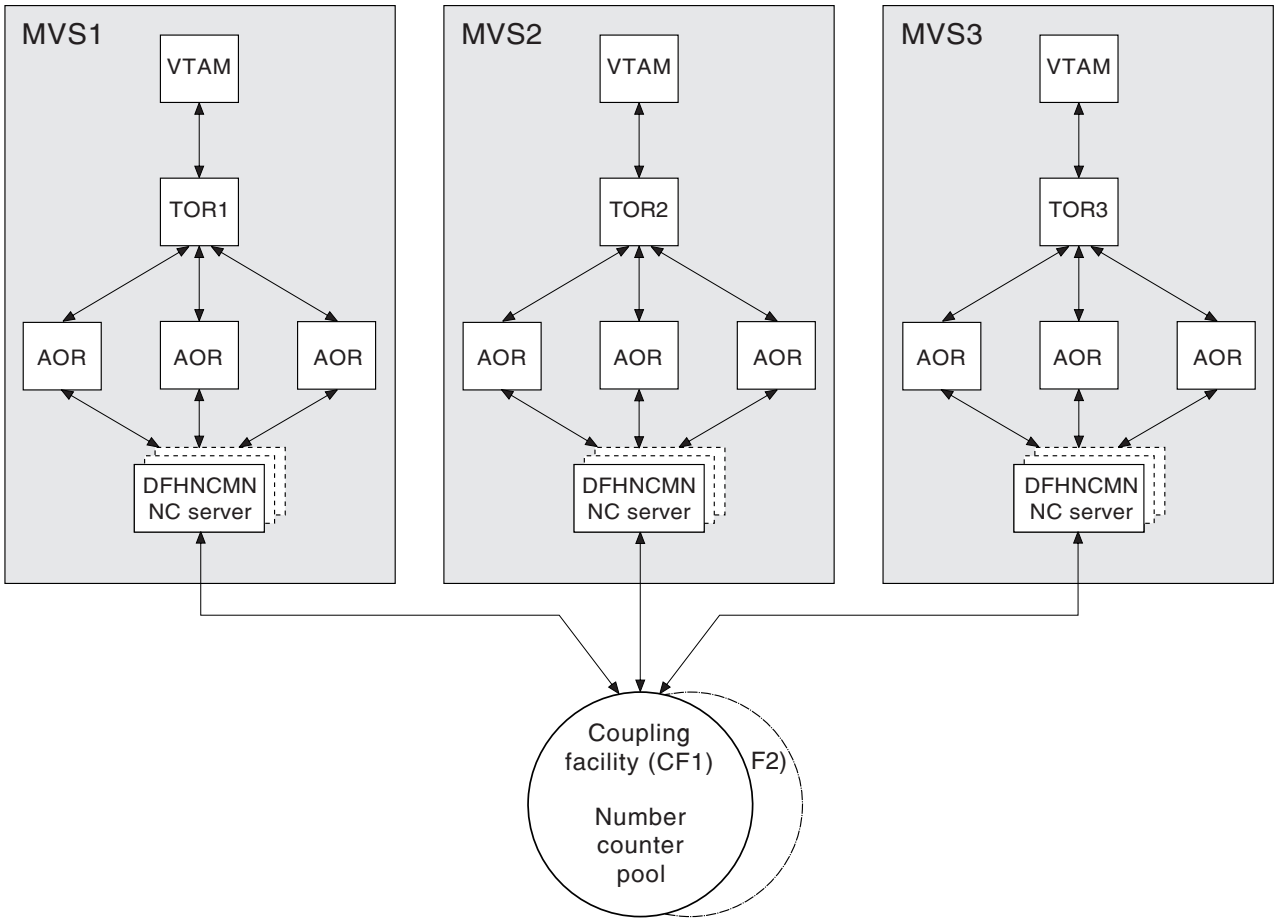


Figure 62. Conceptual view of a Parallel Sysplex with coupling facility data table servers

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, CICSTS13.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 384.

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          CICS CFDT Server
//STEPLIB  DD   DSN=CICSTS13.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
/*
```

Figure 63. 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 384)
- Security parameters (on page 385)
- Statistics parameters (on page 386)
- List structure parameters (on page 386)
- Debug trace parameters (on page 387)
- Tuning parameters (on page 388)
- Lock wait parameters (on page 389)
- Warning parameters (on page 390)
- Automatic structure alter parameters (on page 390)
- Reserved space parameters (on page 391).

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**.

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.

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 399). 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 **ELEMRATIO**.

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 393.

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 later 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 392.

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 395 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 396.

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
Terminate the server immediately.

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 384 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**.

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**.

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 bytes x number of entries) + (100 bytes x 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.

Unload JCL example

```
//UNLDCFD1 JOB  ...
//DTUNLOAD EXEC PGM=DFHCFMN        CICS CF data table server program
//STEPLIB  DD   DSN=CICSTS13.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
/*
```

Reload JCL example

```
//RELDCFD1 JOB  ...
//DTRELOAD EXEC PGM=DFHCFMN        CICS CF data table server program
//STEPLIB  DD   DSN=CICSTS13.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
/*
```

Chapter 28. Starting a named counter server

This chapter describes how to start up a named counter server, and provides information about the following:

- "Overview of a named counter server"
- "Defining and starting a named counter server region" on page 404
- "Controlling named counter server regions" on page 408
- "Named counter server parameters" on page 406.

Note: Before you can start a server for named named counter pool, first define the coupling facility structure to be used for the pool. See "Chapter 8. Defining sequence numbering resources" on page 75 for information about defining a coupling facility list structure for a named counter server.

Overview of a named counter server

The CICS named counter facility provides a facility for generating unique sequence numbers for use by application programs in a Parallel Sysplex environment. Each named counter is held in a pool of named counters, which resides in a coupling facility list structure. Retrieval of the next number in sequence from a named counter is through a callable programming interface.

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 DFHNCLS, is specified in the start-up JCL for the server.

Figure 64 on page 404 illustrates a Parallel Sysplex with three CICS AORs linked to named counter servers.

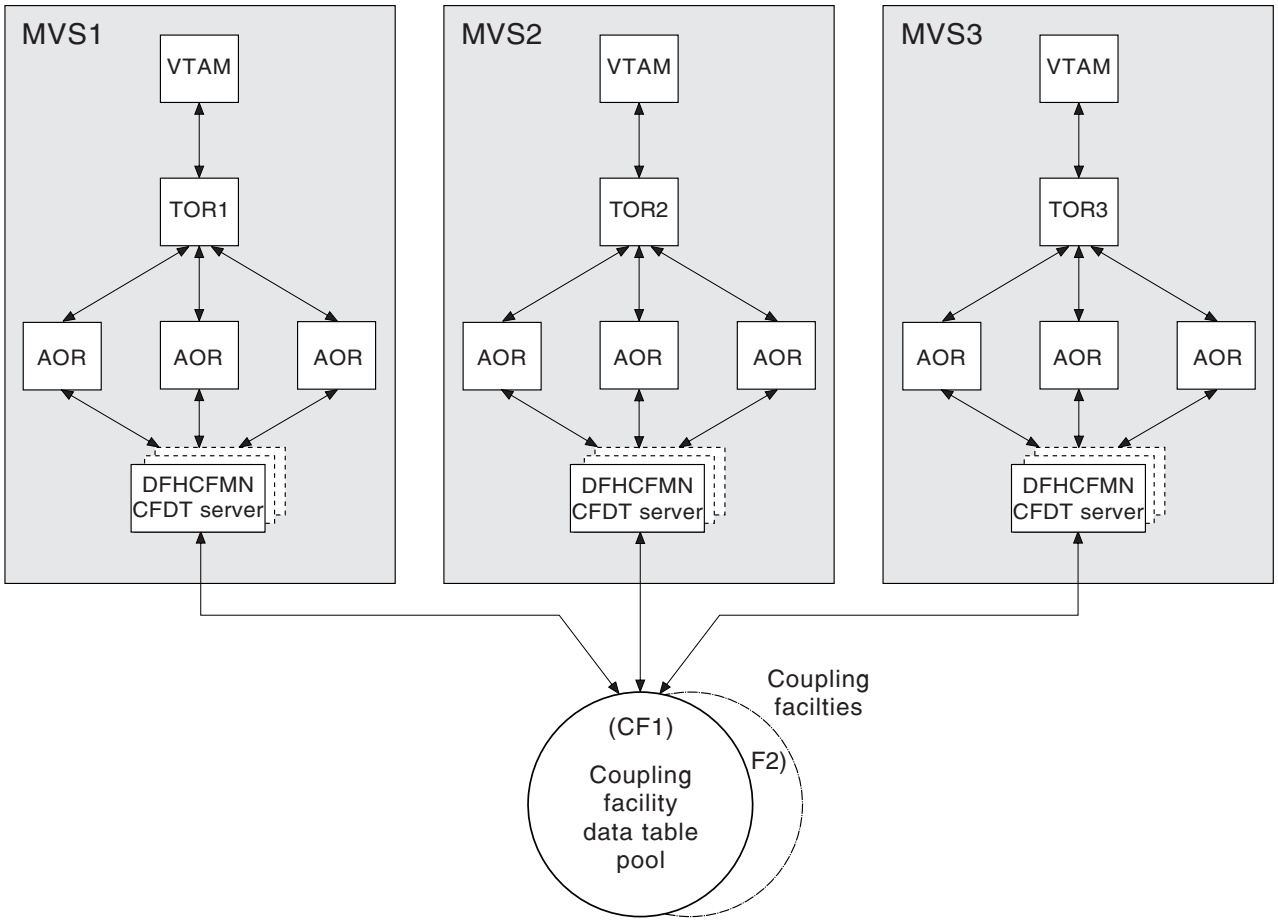


Figure 64. Conceptual view of a Parallel Sysplex with named counter servers

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, CICSTS13.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 406.

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 data table 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

```
//MVSnNC1 JOB  ...
//NCSERVER EXEC PGM=DFHNCMN,REGION=32M       Named counter server
//STEPLIB  DD   DSN=CICSTS13.CICS.SDFHAUTH,DISP=SHR  Authorized library
//SYSPRINT DD   SYSOUT=*                        Messages and statistics
//SYSIN    DD   *
POOLNAME=MVSnNC1                          Pool name
/*
```

Figure 65. 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 DFHNCLS to create its own server name, as in DFHNC.*poolname*, and also to the prefix DFHCFLS_ to create the name of the coupling facility list structure, as in DFHCF_*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**.

List structure parameter

This 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" 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 410.

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
Terminate the server immediately.

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 406 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:

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 **PARM** or **PARMS**.

ALLPARAMETERS
Display all parameter values, which are those listed for PARAMETERS, above, plus:

CFTRACE
RQTRACE
ENTRYWARN
ENTRYWARNINC

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**.

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=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.

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=CICSTS13.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
/*
```

Reload JCL example

```
//RELDNCD1 JOB  ...
//NCRELOAD EXEC PGM=DFHNCMN        CICS named counter server program
//STEPLIB  DD   DSN=CICSTS13.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
/*
```

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*.

Part 5. Appendixes

Appendix. System initialization parameters grouped by functional area

The following table is provided as a guide to system initialization parameters, related by functional group (for example, XRF or intersystem communication). By using this table as a guide, you should find it easier to ensure that you code **all** the parameters that are needed for a particular function you require in CICS.

Note: A check mark (✓) in column two indicates that the parameters are read from the SIT directly by the CICS component that uses them, and are not obtained through the parameter manager domain interface. For more information about the parameter manager domain, see “The CICS parameter manager domain” on page 319.

Table 39. System initialization parameters grouped by functional area

Functional group		System initialization keywords
Application considerations	✓	CMDPROT, CWAKEY, INITPARM, LGNMSG, OPERTIM, TCTUALOC, TCTUAKEY
Autoinstall for VTAM terminals and APPC connections	✓	AIEXIT, AILDELAY, AIQMAX, AIRDELAY, AICONS
Autoinstall for programs		PGAICTLG, PGAIEXIT, PGAIPGM
Basic mapping support	✓	BMS, PGCHAIN, PGCOPY, PGPURGE, PGRET, PRGDLAY, SKRxxxx
Data interchange	✓	DIP
Dispatcher functions		ICV, ICVTSD, MXT, PRTYAGE, SUBTSKS
Dump functions		DUMP, DUMPDS, DUMPSW, DURETRY, SYDUMAX, TRDUMAX TRTRANSZ, TRTRANTY
Exits	✓	TBEXITS, TRAP
Extended recovery facility	✓	ADI, AUTCONN, CLT, JESDI, PDI, RMTRAN, RST, TAKEOVR, XRF, XRFSOFF, XRFSTME
Files (user)		FCT, FTIMEOUT, RLS
Front end programming interface		FEPI
Intersystem communication and multiregion operation	✓	APPLID, DTRPGM, DTRTRAN, DTRPGM, IRCSTRT, ISC, MROFSE, MROBTCH, MROLRM, SYSIDNT (For ISC, see also VTAM group.)
Journaling	✓	AKPFREQ
Loading programs		LLACOPY, LPA, PGAICTLG, PGAIPGM, PGAIEXIT, PLTPI, PLTPISEC, PLTPIUSR, PRVMOD
Miscellaneous	✓	DATFORM, DB2CONN, DBCTLCON, DOCCODEPAGE, FLDSEP, FLDSTRT, ISRDELAY, MSGCASE, MSGLVL, NATLANG, PRINT, SPOOL, STATRCD, WEBDELAY

Table 39. System initialization parameters grouped by functional area (continued)

Functional group		System initialization keywords
Monitoring		MCT, MN, MNCONV, MNFREQ, MNSUBSYS, MNSYNC, MNTIME, MNEVE, MNEXC, MNPER
RDO: file control attributes for the CICS system definition data set		CSDACC, CSDBKUP, CSDBUFND, CSDBUFNI, CSDDISP, CSDDSN, CSDFRLOG, CSDINTEG CSDJID, CSDLSRNO, CSDRECOV, CSDRLS CSDSTRNO
Security	✓	CMDSEC, CMDPROT, CONFDATA, CONFTXT, DFLTUSER, ESMEXITS, KEYFILE, PLTPISEC, PLTPIUSR, PSBCHK, RESSEC, SEC, SECPRFX, SSL, XAPPC, XCMD, XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, XTRAN, XTST, XUSER
Signon		SNSCOPE, USRDELAY. (See also Security, and Terminal and LU management.)
Storage management		CDSASZE, CHKSTRM, CHKSTSK, CMDPROT, CWAKEY, DSASLIM, ECDSASZE, EDSALIM, ERDSASZE, ESDSASZE, EUDSASZE, RDSASZE, RENTPGM, SDSASZE, STGPROT, STGRCVY, TCTUALOC, TCTUAKEY, TRANISO, UDSASZE
Supervisor calls	✓	CICSSVC, SRBSVC
System initialization		GRPLIST, INITPARM, NEWSIT, OFFSITE, PARMERR, PLTPI, PLTPIUSR, PLTPISEC, SIT, START, STARTER, SUFFIX, WRKAREA
System recovery	✓	SRT
System termination	✓	PLTSD, SDTRAN, XLT
Temporary storage	✓	TS, TST
Terminal and LU management	✓	APPLID, CLSDSTP, DSHIPIDL, DSHIPINT, EODI, FLDSEP, FLDSTRT, GMTEXT, GMTRAN, GNTRAN, GRNAME, HPO, ICVTSD, LGNMSG, OPERTIM, OPNDLIM, PRINT, PSDINT, PSTYPE, PVDELAY, RAMAX, RAPOOL, RESP, TCAM, TCP, TCSACTN, TCSWAIT, TCT, TCTUALOC, TCTUAKEY, VTAM (See also autoinstall for VTAM terminals.)
Trace		AUXTR, AUXTRSW, INTTR, GTFTR, TRTABSZ, SPCTR, SPCTRxx, STNTR, STNTRxx, SYSTR, TDINTRA, USERTR
Transient data	✓	DCT, TD
Timer		ICP, ICVR, OPERTIM

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

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 request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5655-147

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1682-02

Spine information:

CICS TS for OS/390 CICS System Definition Guide Release 3