

TXSeries for Multiplatforms



CICS Administration Guide for Open Systems

Version 6.0

TXSeries for Multiplatforms



CICS Administration Guide for Open Systems

Version 6.0

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 323.

First Edition (November 2005)

This edition replaces SC09-4587-04.

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

© Copyright International Business Machines Corporation 2001, 2005. All rights reserved.

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

Contents

Figures	ix

Tables	xi

About this book	xiii
Who should read this book	xiii
Document organization	xiii
Conventions used in this book.	xiv
How to send your comments	xvi

Part 1. Configuring, modifying and using your CICS region 1

Chapter 1. Understanding CICS administration tasks and tools 3

Understanding the classification of resource definitions	5
Identifying resource definition classes	5
Comparing resource definitions across CICS family members	8
Using shorthand in the resource definitions	8
Tools for configuring CICS regions and resources	9
The CICS control program commands	9
The CICS resource management commands	9
Other commands used to administer a CICS system	10
The AIX System Management Interface Tool	10

Chapter 2. Enabling administrator, developer, and end user access to CICS. 11

Setting up a CICS system for an administrator.	11
Deciding where to set your environment variables	13
Considering setup requirements for CICS developers.	13
Setting up a CICS system for an end user	14
Adding a new user to the region	14
Using CICS facilities to change a password.	15
Setting the language for the user's client connection	15

Chapter 3. Configuring and modifying CICS regions 17

Creating a region by using the CICS control program command	17
Examining the effects of using the cicscp command to set up a region.	18
Deleting a region by using the cicscp command	19
Creating a region by using the CICS resource management commands	20
Changing the Region Definitions (RD) attributes of a new region	20
Setting the language for the region	21

Setting up a separate file system for the CICS log directory	22

Chapter 4. Working with a file manager in a CICS region 23

Administration of file and queue management on an SFS server	23
How a CICS system uses an SFS file manager	24
Overriding the defaults imposed by the cicscp start sfs_server command.	29
Enlarging the logical volumes for the SFS server	30
Using nlsString data types in SFS files	30
Deleting an SFS server by using the cicscp command	31
Administration of file and queue management on DB2	31
How a CICS system uses a DB2 file manager	32
Configuring a DB2 database.	34
Preparing for DB2 file and queue management	37
Improving DB2 performance	38
Security considerations for DB2 file management	40
Administration of file and queue management on Oracle	40
How a CICS system uses an Oracle file manager	41
Configuring an Oracle database	44
Preparing for Oracle file and queue management	47
Improving Oracle performance	49
Security considerations for Oracle file management	49
Migrating file and queue management (DB2 and SFS).	49
Moving file management from SFS to DB2	50
Moving file management from DB2 to SFS	52
Migrating file and queue management (Oracle and SFS).	53
Moving file management from SFS to Oracle	54
Moving file management from Oracle to SFS	56

Chapter 5. Working with CICS queues 59

Creating the resource definitions for adding queues to the SFS server	59
Creating the resource definitions for adding queues to the DB2 file manager	62
Configuring resources and DB2 with cicsdb2conf	63
Creating the resource definitions for adding queues to the Oracle file manager	64
Configuring resources and Oracle with cicsoraconf	64
Setting up temporary storage queues (TSQ)	66
About temporary storage queues (TSQ)	66
Configuring the Temporary Storage Definitions (TSD)	67
Setting up transient data queues (TDQs).	68
About transient data queues (TDQs)	68
Configuring the Transient Data Definitions (TDD)	69

Chapter 6. Working with CICS clients	71
Using the Telnet client (cicsteld)	71
Telnet terminals and the cicsteld server	72
Setting up the cicsteld server for use in DBCS or MBCS environments	76
Using Telnet to connect a client to a region	77
Destroying a Telnet server configuration	77
Behavior of cicsteld	77
Using the ECI and EPI applications for client access	78
Using the IBM CICS Universal Client products	78
Using the Communication Definitions (CD) autoinstall program	80
Understanding how the listener process works	81
Configuring the region for IBM CICS Universal Client products	82
Setting up Terminal Definitions (WD)	83
Selecting a model type when the cicsteld server is used	83
Selecting a model type when an IBM CICS Universal Client product is used	84
The CICS-supplied autoinstall models	85
Administering Terminal Definitions (WD)	87
 Chapter 7. Ensuring security in a CICS environment	 89
Authenticating user access to CICS	89
Overview of CICS user authentication	89
Security considerations in an XA-enabled environment	90
Using an External Authentication Manager	91
How to use an EAM with CICS	91
Defining the interface between CICS and an EAM	91
Defining an EAM at startup	94
Authorizing access to CICS transactions	94
Accessing user-defined transactions	94
An example of running a secure transaction	96
Authorizing access to CICS-supplied transactions	96
Authorizing access to the CEDF debugging transaction	97
Managing transaction security	98
Authorizing user access to CICS resources	99
Access from outside the runtime	99
Authorizing access to startup and shutdown programs	99
Authorizing access to resource definitions with CICS commands	99
Authorizing access to resources outside the running region	100
Authorizing access to CICS resources from applications	100
Using the TSLKeyList and RSLKeyList attributes	102
More about resource security	102
Resource security checklist	106
Using an External Security Manager	106
How to use an ESM with CICS	106
Defining the interface between CICS and an ESM	107
Defining an ESM at startup	111
Defining CICS to an ESM	111

Using the ESM with CICS system transactions	111

Part 2. Setting up resources for use within a CICS region 113

Chapter 8. Working with CICS resource definitions 115

Understanding how the resource definition databases are used	115
Using CICS resource management commands to update resource definitions	116
Understanding the scope of commands affecting resource definitions	116
Updating resource definitions in a running region	116
Using CEMT to manage runtime resources	119
Using the EXEC CICS INQUIRE and the EXEC CICS SET commands	123
Identifying exceptions to when resource definitions can be changed	123
Defining resources to be selectively installed	124
Configuring for shared resource definitions	125
Sharing Region Definitions (RD)	125
Sharing SFS Region Definitions (RD)	125
Sharing Communications Definitions (CD)	125
Sharing definitions that can be function shipped	126

Chapter 9. Working with Program Definitions and Transaction Definitions 127

Configuring Program Definitions (PD)	127
Configuring the Program Definitions (PD)	127
Working with PD in the runtime database	128
Program Definitions and setting up Basic Mapping Support for different languages	128
Autoinstalling Program Definitions (PD)	129
Configuring Transaction Definitions (TD)	130
Configuring the Transaction Definitions (TD) in the permanent database	131
Working with TD in the runtime database	131
Configuring for the in-doubt condition	132

Chapter 10. Working with application data files in CICS. 133

Configuring the File Definitions (FD)	133
Procedures used when working with data files	133
Creating File Definitions (FD) for record types	134
Configuring the File Definitions (FD) in the permanent and runtime databases	135
Working with FD in the runtime database	136
Adding files to the file manager by using Schema File Definitions (SCD)	136
Adding files to the file manager by using the diagnostic tools	139

Chapter 11. Configuring CICS for use with relational databases 143

The CICS-provided database sample programs	143

Using relational databases with CICS	144
Backup and recovery in XA-enabled environments	144
Relay of catastrophic diagnostic information	144
Optimizing the single-phase commit protocol	145
Enabling database connections in XA-enabled environments	146
Preparing the region for use with the RM	147
Setting up Product Definitions (XAD)	147
Enabling Micro Focus Server Express COBOL or ACUCOBOL-GT for use with the RM	149
SQL database setup procedures	149
Using CICS with DB2	150
Using CICS with Informix	153
Using CICS with Oracle	156
Using CICS with Sybase versions 11 and 12	163
Configuring ACUCOBOL-GT for use with relational databases	174

Chapter 12. Preparing CICS for use with Micro Focus Server Express COBOL 175

Setting up the Micro Focus Server Express COBOL runtime environment	175
When to use cicsmkcobol	175
Micro Focus Server Express COBOL programs and relational databases	175
Micro Focus Server Express COBOL license management	176

Chapter 13. Preparing CICS for use with ACUCOBOL-GT 177

Setting up the ACUCOBOL-GT runtime environment	177
When to relink the ACUCOBOL-GT runtime and create cicsprACUCOB	177
ACUCOBOL-GT programs and relational databases	177
ACUCOBOL-GT license management	178

Chapter 14. Working with WebSphere MQ 179

Installation	179
Building your application	179
Using CICS with WebSphere MQ	180
Prerequisite tasks and conditions	180
Procedure for setting up the CICS interface to WebSphere MQ	180
Signal handling when working with WebSphere MQ	183

Chapter 15. Customizing CICS with user exits and user programs 185

Working with CICS user exits	185
Implementing user exits	186
The user programs	188
Using the External Authentication Manager user program	189
Using the External Security Manager user program	189

Using the terminal autoinstall user program	189
Using the performance monitoring user program	190
Using post-initialization user programs	190
Using shutdown user programs	191

Part 3. Maintaining your CICS region 195

Chapter 16. Starting up a CICS system 197

The startup sequence	197
Cold starts and warm starts	198
Cold starting the SFS server	198
Using the CICS control program commands for startup	199
Startup procedures	199
Starting an SFS server by using the cicscp start sfs_server command	199
Starting the region by using the cicscp start region command	199
Starting CICS by using the cicscp start all command	200
Recovery and restart for CICS startup	200
Recovering resource definitions at startup	200
Restarting CICS	200
Recovery control processing	201
Backout processing	201
File manager unavailable at startup and while running transactions	201
Effect of an unavailable file manager at CICS startup	201
Effect of an unavailable file manager at various stages	204
Effect of other unavailable file managers	205

Chapter 17. Shutting down a CICS system 207

The shutdown sequence	207
CICS shutdown processes	207
Normal shutdown	207
Immediate shutdown	208
Cancel (forced abnormal) shutdown	209
Software-initiated shutdown	209
Uncontrolled shutdown	209
Statistics produced at shutdown	209
Printing a software-initiated shutdown dump	210
Shutting down CICS by using cicscp	210
Shutting down the region by using cicscp stop region	210
Shutting down an SFS server by using cicscp stop sfs_server	211
Shutting down CICS by using cicscp stop all	211
Shutting down the region by using CEMT PERFORM SHUTDOWN	211
Shutting down the region by using EXEC CICS PERFORM SHUTDOWN	212

Chapter 18. Monitoring your CICS system 213

Collecting statistics	213
Information collected for CICS statistics	213
How to collect statistics and format a report	214
Setting up Journal Definitions (JD)	217
Configuring the Journal Definitions (JD)	218
Monitoring performance.	219
Standard CICS information.	219
Information from CICS application programs	219
What you must define in the Monitoring Definitions (MD)	220
Setting up Monitoring Definitions (MD)	220
How to control CICS monitoring.	220
Processing of CICS monitoring-facility output	221
Tracing CICS execution performance	221
General comments on using auxiliary tracing	222

Chapter 19. Improving the performance of your CICS system . . . 223

Tuning a CICS system	223
Determining acceptable tuning trade-offs	223
Making changes to the system.	223
Reviewing the results of the tuning	224
Measuring performance	224
Performance measurement tools provided by the operating system	224
Performance measurement tools provided by CICS	225
Performance measurement tools provided by relational databases	225
Symptoms and solutions for CICS performance constraints	225
Symptoms and solutions for storage stress conditions and memory shortages	226
Symptoms and solutions for poor response time	229
Performance considerations for system administrators	230
Improving performance by examining startup and shutdown processes.	230
Improving DB2 performance by tuning parameters	231
Improving performance by adjusting settings for file control	232
Improving performance by uninstalling inactive clients.	233
Improving performance by controlling the rate of requests to the RPC listener process	233
Improving performance by controlling the number of transactions running in a region	234
Improving performance by tuning I/O access	235
Improving performance by balancing throughput and response time.	236
Improving performance by using an SP2 on the AIX platform	236
Performance considerations shared by administrators and developers.	236
Improving performance by careful use of CICS facilities	236
Distributing the workload between regions	237
Possible effects of changing the system clock	238

Chapter 20. Understanding CICS backup and recovery 239

Protecting your CICS system and data	239
Mechanisms that aid backup, recovery, and restart	239
Synchronization points (syncpoints)	239
Two-phase commit	240
CICS recoverable resources.	240
Automatic start.	241
Preparing your CICS systems for recovery.	241
Enabling recovery of the SFS server	242
Enabling recovery of DB2	247
Enabling recovery of Oracle	249
Enabling recovery of CICS	249
Enabling recovery of the CICS subsystem entries	251
Enabling recovery of the CICS user IDs	251
Full recovery after a complete system failure	251
Suggestions for backing up your systems	252
Improving your ability to restore services	253
Scheduling offline backups	255

Part 4. Appendixes 257

Appendix A. Using the AIX System Management Interface Tool (SMIT) . . . 259

Metacharacter restrictions	260
Using SMIT to add resource definitions	260
Using SMIT to add a user	261
Using SMIT to change a resource definition	261
Using SMIT to modify a user	262
Using SMIT to view a resource definition	262
Using SMIT to delete a resource definition	263
Using SMIT to refresh the region with new resource definitions	263
Using SMIT fast path to administer resources	264
Using the SMIT fast path to add a resource	264
Using SMIT fast path to change a resource definition.	265
Using SMIT fast path to inquire about a resource	265
Using SMIT fast path to delete a resource definition.	265
Using SMIT fast path to install a group of resources	265
Using SMIT to add a region	265
Using SMIT to delete a region.	266
Using SMIT to add an SFS server.	266
Using SMIT to add a PPC Gateway server.	267

Appendix B. Using CICS resource management commands for system management. 269

The CICS resource management commands used by the cicscp command	269
Commands used to set up a CICS system	270
Commands used to prepare a region for production	270
Combining the CICS resource management commands to set up CICS	271
Creating the region's default user ID	272

Using the cicsdefault command to define a region	272
Defining a region using the cicsexport and cicsimport commands	273
Creating the user ID for the SFS server	273
Preparing the physical storage devices for the SFS server	274
Defining an SFS server by using the cicssfscree command	275
Starting an SFS server by using the cicssfscree command	276
Setting up the queues by using the cicssfscree command	276
Setting up the queues by using sfsadmin commands	277
Starting a region from the command line	280
Shutting down the region from the command line	281
Shutting down an SFS server by using the cicssfsshut command	282

Using the cicsdestroy command to delete a region	282
Using the cicssfssdestroy command to delete an SFS server	283

Appendix C. Performing CICS administrative functions as a non-root user	285

Glossary	287

Notices	323
Trademarks and service marks	324

Index	327

Figures

1. Region configuration options set up by using the cicscp command 18
2. An example of an SFS file management system for CICS 24
3. Setting NLS data types for SFS files 31
4. An example of a DB2 file management system for CICS 32
5. An example of an Oracle file management system for CICS 42
6. An example of a Telnet session in CICS 72
7. An example of an IBM CICS Universal Client session 79
8. Terminal selection process for a client using Telnet server for CICS 84
9. CEMT INQ 121
10. The CICS implementation of the AP, RM, TM model 148
11. Media Recovery Archiving (MRA) used in combination with mirroring for SFS backup . . 245
12. The Customer Information Control System (CICS) Version 6.0 panel 259

Tables

1. Administration Guide road map	xiii	20. Names of TDQ files and indices for a CICS region that is using DB2	69
2. Conventions that are used in this book	xiv	21. Names of TDQ files and indices for a CICS region that is using Oracle	69
3. Resource definition types	8	22. Locations for the inetd daemon scripts and variables requiring customization	74
4. Environment variables used for CICS administration on the AIX platform	12	23. Location of the default C source and executable code for DFHCCINX.	80
5. Environment variables used for CICS administration on the HP-UX platform	12	24. CICS-supplied autoinstall models	85
6. Environment variables used for CICS administration on the Solaris platform	13	25. Required permissions for CICS product files	100
7. Where to set environment variables on the AIX, HP-UX, and Solaris platforms.	13	26. RSL key values	103
8. Location of the client message catalogs	15	27. Default ESM parameter field lengths.	110
9. NLSPATH environment variable settings	16	28. Library path environment variables	111
10. The location of the README file on the AIX, HP-UX, and Solaris platforms	49	29. Location of the cicsmkcobol program	149
11. Queue name values (where %R is the region name)	51	30. Location of the CICS system /bin directory	175
12. The location of the README file on the AIX, HP-UX, and Solaris platforms	53	31. User exit numbers	186
13. Queue name values (where %R is the region name)	55	32. Effect of an unavailable file manager at CICS startup.	201
14. Table names for CICS queues in DB2	64	33. Effect of an unavailable file manager at various stages of CICS processing.	204
15. Table names for CICS queues in Oracle	66	34. Effect of unavailability of other file managers	206
16. Names of TSQ files and indices for a CICS region that is using an SFS server	67	35. CICS response time checklist	229
17. Names of TSQ files and indices for a CICS region that is using DB2	67	36. Performance cost of starting up CICS	230
18. Names of TSQ files and indices for a CICS region that is using Oracle	67	37. CICS response time checklist	234
19. Names of TDQ files and indices for a CICS region that is using an SFS server	69	38. Location of example files for testing the validity of SFS data	247
		39. CICS system management tasks and the CICS commands to complete them	271

About this book

This guide describes how to customize and operate CICS® on the AIX®, HP-UX, and Solaris operating systems. The information in this guide assumes that your CICS system has been installed and tested according to the procedures described in the *TXSeries for Multiplatforms Installation Guide*. It does not describe how to install CICS.

Who should read this book

This book is for system administrators who are responsible for customizing and operating CICS.

Familiarity with administration of your operating system is helpful because many of the system administration procedures are used with CICS.

If you are using a DB2® or Oracle database for file management, then you need to know about database administration for your database.

Document organization

Table 1. Administration Guide road map

If you want to...	Refer to...
Read about the conventions used in this book.	"Conventions used in this book" on page xiv.
Get an overview of the types of tasks and tools required to administer a CICS system.	Chapter 1, "Understanding CICS administration tasks and tools," on page 3.
Enable system access for different types of CICS users	Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.
Set up and customize your CICS system's regions.	Chapter 3, "Configuring and modifying CICS regions," on page 17.
Set up and customize your CICS system's file manager.	Chapter 4, "Working with a file manager in a CICS region," on page 23.
Set up and customize your CICS system's queues.	Chapter 5, "Working with CICS queues," on page 59.
Set up and customize your CICS system's clients.	Chapter 6, "Working with CICS clients," on page 71.
Protect your system from unauthorized access.	Chapter 7, "Ensuring security in a CICS environment," on page 89.
Set up and customize your CICS system's resource definitions.	Chapter 8, "Working with CICS resource definitions," on page 115.
Enable program and transaction interaction with CICS.	Chapter 9, "Working with Program Definitions and Transaction Definitions," on page 127.
Set up and customize your CICS system's application data files.	Chapter 10, "Working with application data files in CICS," on page 133.
Make CICS work with relational database management products.	Chapter 11, "Configuring CICS for use with relational databases," on page 143.

Table 1. Administration Guide road map (continued)

If you want to...	Refer to...
Make CICS work with Micro Focus Server Express COBOL applications.	Chapter 12, "Preparing CICS for use with Micro Focus Server Express COBOL," on page 175.
Make CICS work with ACUCOBOL-GT applications.	Chapter 13, "Preparing CICS for use with ACUCOBOL-GT," on page 177.
Make CICS work with WebSphere® MQ.	Chapter 14, "Working with WebSphere MQ," on page 179.
Make CICS work with user applications.	Chapter 15, "Customizing CICS with user exits and user programs," on page 185.
Start up or shut down your CICS system.	Chapter 16, "Starting up a CICS system," on page 197, and Chapter 17, "Shutting down a CICS system," on page 207.
Monitor your system for performance.	Chapter 18, "Monitoring your CICS system," on page 213.
Tune your system for performance.	Chapter 19, "Improving the performance of your CICS system," on page 223.
Prepare CICS for backup and recovery or recover CICS after a system failure.	Chapter 20, "Understanding CICS backup and recovery," on page 239.
Administer your AIX system by using the AIX system administration tool.	Appendix A, "Using the AIX System Management Interface Tool (SMIT)," on page 259.
Set up a region by using the CICS resource management commands.	Appendix B, "Using CICS resource management commands for system management," on page 269.
Read about the CICS library and related books.	See the <i>TXSeries for Multiplatforms Concepts and Planning</i> .

Conventions used in this book

TXSeries® for Multiplatforms documentation uses the following typographical and keying conventions.

Table 2. Conventions that are used in this book

Convention	Meaning
Bold	Indicates values that you must use literally, such as commands, functions, and resource definition attributes and their values. When referring to graphical user interfaces (GUIs), bold also indicates menus, menu items, labels, buttons, icons, and folders.
Monospace	Indicates text that you must enter at a command prompt. Monospace also indicates screen text and code examples.
<i>Italics</i>	Indicates variable values that you must provide (for example, you supply the name of a file for <i>file_name</i>). Italics also indicates emphasis and the titles of books.
< >	Encloses the names of keys on the keyboard.
<Ctrl- <i>x</i> >	Where <i>x</i> is the name of a key, indicates a control-character sequence. For example, <Ctrl-c> means hold down the Ctrl key while you press the c key.
<Return>	Refers to the key labeled with the word Return, the word Enter, or the left arrow.

Table 2. Conventions that are used in this book (continued)

Convention	Meaning
%	Represents the UNIX [®] command-shell prompt for a command that does not require root privileges.
#	Represents the UNIX command-shell prompt for a command that requires root privileges.
C:\>	Represents the Windows [®] command prompt.
>	When used to describe a menu, shows a series of menu selections. For example, "Select File > New " means "From the File menu, select the New command."
Entering commands	When instructed to "enter" or "issue" a command, type the command and then press < Return >. For example, the instruction "Enter the ls command" means type ls at a command prompt and then press < Return >.
[]	Encloses optional items in syntax descriptions.
{ }	Encloses lists from which you must choose an item in syntax descriptions.
	Separates items in a list of choices enclosed in { } (braces) in syntax descriptions.
...	Ellipses in syntax descriptions indicate that you can repeat the preceding item one or more times. Ellipses in examples indicate that information was omitted from the example for the sake of brevity.
IN	In function descriptions, indicates parameters whose values are used to pass data to the function. These parameters are not used to return modified data to the calling routine. (Do <i>not</i> include the IN declaration in your code.)
OUT	In function descriptions, indicates parameters whose values are used to return modified data to the calling routine. These parameters are not used to pass data to the function. (Do <i>not</i> include the OUT declaration in your code.)
INOUT	In function descriptions, indicates parameters whose values are passed to the function, modified by the function, and returned to the calling routine. These parameters serve as both IN and OUT parameters. (Do <i>not</i> include the INOUT declaration in your code.)
\$CICS	Indicates the full path name of the location in which the CICS product is installed; for example, <i>/usr/lpp/cics</i> on AIX. If the CICS environment variable is set to the product path name, you can use the examples exactly as shown in this book; otherwise, you must replace all instances of \$CICS with the CICS product path name.
CICS on Open Systems	Refers collectively to the CICS product for all supported UNIX platforms.
TXSeries for Multiplatforms	Refers collectively to the CICS for AIX, CICS for HP-UX, CICS for Solaris, and CICS for Windows products.
CICS	Refers generically to the CICS for AIX, CICS for HP-UX, CICS for Solaris, and CICS for Windows products. Other CICS products in the CICS Family are distinguished by their operating system (for example, IBM [®] mainframe-based CICS for the ESA, MVS [™] , and VSE platforms).

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this book or any other TXSeries for Multiplatforms documentation, send your comments by e-mail to idrcf@hursley.ibm.com. Be sure to include the name of the book, the document number of the book, the version of TXSeries for Multiplatforms, and, if applicable, the specific location of the information you are commenting on (for example, a page number or table number).

Part 1. Configuring, modifying and using your CICS region

This part of the guide provides information to help you configure your CICS system in the following conditions:

- You have completed installation and initial configuration of your system, as described in the *TXSeries for Multiplatforms Installation Guide*, and want to further customize the default system that you have created.
- You need to create additional CICS regions or SFS servers, or to modify the setup of existing ones.
- You need to make changes to the settings for a system to reflect updates or additions to its configuration.

The topics include:

- Chapter 1, “Understanding CICS administration tasks and tools,” on page 3
- Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11
- Chapter 3, “Configuring and modifying CICS regions,” on page 17
- Chapter 4, “Working with a file manager in a CICS region,” on page 23
- Chapter 5, “Working with CICS queues,” on page 59
- Chapter 6, “Working with CICS clients,” on page 71
- Chapter 7, “Ensuring security in a CICS environment,” on page 89

Chapter 1. Understanding CICS administration tasks and tools

The administration of a CICS system is achieved by performing several discrete tasks. This guide identifies the tasks that you need to perform to customize your CICS system, and it describes the tools that you need when you do these tasks.

This guide assumes that you have already set up the operating system and that you have completed the initial installation, configuration and startup tasks for your CICS system according to the procedures that are described in the *TXSeries for Multiplatforms Installation Guide*.

Assuming that you have completed the tasks that are described in the *TXSeries for Multiplatforms Installation Guide*, this is the status of your CICS system:

- You have used operating system commands that are appropriate to your platform to create the file volumes that your operating system uses.
- You have created your CICS users (**root** and **cics**) and your CICS groups (**cics** and **cicsterm**).
- You have designated the user **root** as the CICS system administrator.
- You have set up either a Structured File Server (SFS) server or a DB2 file system as your file management system.
- You have configured a basic region, by using the default settings that the **cicscp** command supplies.
- You have created a client, enabled 3270 keyboard mapping, and used your client to connect to the region.

After the initial setup of the operating system and the CICS system, the focus of a system administrator's job moves to the running of the CICS system; that is, monitoring system activity, improving performance, ensuring security, and enabling backup and recovery.

Administration tasks include the addition of resources and users to customize your CICS system's growth in response to your business needs.

This guide organizes discussion about the CICS system administration tasks into three parts and appendixes:

- Some common administrative procedures and expected system responses are discussed in Part 1, "Configuring, modifying and using your CICS region," on page 1. They include the following topics:
 - Becoming familiar with the CICS resource definitions classifications and how they are used to create and customize resources within a CICS system. See "Understanding the classification of resource definitions" on page 5.
 - Becoming familiar with the command tools that enable configuration and administration on your CICS system. See "Tools for configuring CICS regions and resources" on page 9.
 - Ensuring that a CICS system administrator exists with the appropriate privileges to maintain the system. Determining how to configure an ID for use by a CICS developer and by a CICS end user. See Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.
 - Creating and modifying CICS regions. See Chapter 3, "Configuring and modifying CICS regions," on page 17.

- Setting up a CICS file manager by using SFS or DB2. See Chapter 4, “Working with a file manager in a CICS region,” on page 23.
- Entering the resource definitions that enable queue access for your CICS system. See Chapter 5, “Working with CICS queues,” on page 59.
- Setting up clients and Telnet users. See Chapter 6, “Working with CICS clients,” on page 71.
- Working with security in your CICS environment. See Chapter 7, “Ensuring security in a CICS environment,” on page 89.
- Expanding your CICS system by enabling it to interact with user applications and third-party products is discussed in Part 2, “Setting up resources for use within a CICS region,” on page 113 of this guide. Specific topics include the following:
 - Becoming familiar with the commands that are used to add, modify, and delete CICS resource definitions within a CICS system. See Chapter 8, “Working with CICS resource definitions,” on page 115.
 - Becoming familiar with the commands that are used to add, modify, and delete Program Definitions and Transaction Definitions within a CICS system. See Chapter 9, “Working with Program Definitions and Transaction Definitions,” on page 127.
 - Adding an application’s data files to the file manager and creating the Program Definitions and Transaction Definitions that identify these resources to CICS. See Chapter 10, “Working with application data files in CICS,” on page 133.
 - Preparing your CICS system for use with relational database products. See Chapter 11, “Configuring CICS for use with relational databases,” on page 143.
 - Configuring CICS for use with Micro Focus Server Express. See Chapter 12, “Preparing CICS for use with Micro Focus Server Express COBOL,” on page 175.
 - Customizing CICS for use with the IBM WebSphere MQ product. See Chapter 14, “Working with WebSphere MQ,” on page 179.
 - Customizing CICS with user exits or user programs. See Chapter 15, “Customizing CICS with user exits and user programs,” on page 185.
- Processes that you need to maintain your CICS system are discussed in Part 3, “Maintaining your CICS region,” on page 195. The following activities are included:
 - Understanding the behavior of your CICS system when you start up and shut down your system resources. See Chapter 16, “Starting up a CICS system,” on page 197 and Chapter 17, “Shutting down a CICS system,” on page 207.
 - Gathering statistics and monitoring the performance of your CICS system. See Chapter 18, “Monitoring your CICS system,” on page 213.
 - Tuning performance parameters. See Chapter 19, “Improving the performance of your CICS system,” on page 223.
 - Ensuring the backup and recovery for your CICS system. See Chapter 20, “Understanding CICS backup and recovery,” on page 239.
- The appendixes for this guide provide alternative methods for performing command administrative tasks. Specific topics include:
 - Guidance on using the AIX administrative tool. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259.

- Guidance on using CICS resource management commands to perform administrative tasks. See Appendix B, “Using CICS resource management commands for system management,” on page 269.
- A glossary is located at the end of the book to provide for a quick reference on terminology and acronyms.

Understanding the classification of resource definitions

A CICS *resource* is any facility or component of a CICS system that is required to perform a task. To be used by CICS, each resource requires a resource definition. “Identifying resource definition classes” shows the classification of the definable resources that CICS uses. Many resource definitions are automatically set when the region configuration commands are initially issued.

The CICS resource definitions for a particular region are stored on two databases: the *runtime* database and the *permanent* database. The permanent database contains the definitions that are used to define the state of a region and its resources when it is cold started. When a region is cold started, CICS installs permanent definitions into the runtime database. CICS actually runs from the runtime database.

Throughout this document, the discussion of each definition topic includes when the definition is set, its default values, and the commands that are used to modify the definition. This section gives an overview of the resource definition classifications. Chapter 8, “Working with CICS resource definitions,” on page 115 shows how to add, modify, or delete the resource definition.

“Comparing resource definitions across CICS family members” on page 8 shows the names that are used for resource classes on a CICS mainframe, and the names that are used for corresponding resource classes on the distributed platform.

“Using shorthand in the resource definitions” on page 8 shows the shorthand method that you can use within CICS to identify resource definitions.

Identifying resource definition classes

Resource definitions identify all resources to CICS. This list groups the classification of the different types of resources into three categories: resource definitions that affect the whole region, resource definitions that affect file and gateway servers, and resource definitions that affect programs and transactions. The following list contains the classes of definable resources separated into these categories. Detailed information on these classes can be found in the *TXSeries for Multiplatforms Administration Reference*.

Resource Definitions that affect regions

Resource definitions that affect regions create a unique environment for each region. These classifications of definitions allow region configuration, data exchange, and resource usage.

Communications Definitions (CD)

The Communications Definitions (CD) describe the configuration of other systems with which your CICS region can communicate, including other CICS family systems. Each CD entry describes a remote system. See *TXSeries for Multiplatforms Intercommunication Guide* for more information.

File Definitions (FD)

The File Definitions (FD) define data files that are available in the region. You can define remote and local files in the FD. Local FDs provide you

with a way of accessing files that are on any server that is defined in the Structured File Server Definitions (SSD) for this region. See “Configuring the File Definitions (FD)” on page 133 for more information about FD usage.

Gateway Definitions (GD)

The Gateway Definitions (GD) define the PPC gateways to other systems with which the region can communicate. See the *TXSeries for Multiplatforms Intercommunication Guide* for more information.

Journal Definitions (JD)

The Journal Definitions (JD) specify destinations for journaling output from the CICS region. See “Setting up Journal Definitions (JD)” on page 217 for more information on JD usage.

Listener Definitions (LD)

The Listener Definitions (LD) specify methods that a CICS region uses to listen for connection requests that are coming from IBM CICS Universal Client products across TCP/IP, TCP/IP IIOP, or local SNA. See “Understanding how the listener process works” on page 81 for more information on LD usage.

Monitoring Definitions (MD)

The Monitoring Definitions (MD) specify the destination and contents of monitoring output from a CICS region. This is a special type of resource class that consists of a single record. See “Setting up Monitoring Definitions (MD)” on page 220 for more information on MD usage.

Object Definitions (OD)

The Object Definitions (OD) define values that are used for the CICS IIOP ORB. See *TXSeries for Multiplatforms IIOP and Java™ Programming Guide* for more information.

Product Definitions (XAD)

The Product Definitions (XAD) hold information that is needed to interface to other transactional products, for example XA-compliant databases, that use the X/Open XA protocol. See “Setting up Product Definitions (XAD)” on page 147 for more information on XAD usage.

Region Definitions (RD)

The Region Definitions (RD) define all the parameters that are used by the CICS initialization process. The RD includes parameters that are used to initialize a region, and parameters that are used to configure the SFS server.

Region Definitions (RD) for file server

The Region Definitions (RD) for the file server define all the initialization and control parameters that are required for the startup of an SFS server or DB2 database file server. See “Creating the resource definitions for adding queues to the SFS server” on page 59 and “Creating the resource definitions for adding queues to the DB2 file manager” on page 62 for more information about RD usage for file servers.

Temporary Storage Definitions (TSD)

The Temporary Storage Definitions (TSD) define template data identifiers for temporary storage queues. CICS requires TSD entries for security, recoverability, and remoteness. You specify generic data identifiers so that any unique temporary storage identifier (generated dynamically in a program) that begins with the same characters as does a generic identifier (in the TSD) can automatically acquire the same properties that the TSD

entries have. You can define remote and local queues in the TSD. See “Configuring the Temporary Storage Definitions (TSD)” on page 67 for more information on TSD use.

Terminal Definitions (WD)

The Terminal Definitions (WD) define the configuration of terminals that are available in the region, including models for terminals that can be autoinstalled. You can define remote and local terminals in the WD. See “Setting up Terminal Definitions (WD)” on page 83 for more information about WD use.

Transient Data Definitions (TDD)

The Transient Data Definitions (TDD) define transient data queues that are available in a region. You can define remote and local queues in the TDD. See “Configuring the Transient Data Definitions (TDD)” on page 69 for more information about TDD use.

User Definitions (UD)

The User Definitions (UD) define terminal user information. CICS uses this definition to check security for transactions and resources. See “Creating the region’s default user ID” on page 272 for more information about UD use.

Resource definitions that affect file and gateway servers

Resource definitions that affect servers control the structure of files and queues that the servers use. This set of resource definitions is located in the `/var/cics_servers` file system that is used for the servers. These definitions are used to pass parameters to the servers when they are started, and to help in adding data files to the file manager. Definitions in this classification can be set only in the permanent database. No runtime database is associated with these resource definitions.

Gateway Server Definitions (GSD)

The Gateway Server Definitions (GSD) specify the characteristics that are required to start and stop the PPC Gateway server. See the *TXSeries for Multiplatforms Intercommunication Guide* for more information.

Schema File Definitions (SCD)

The Schema File Definitions (SCD) define all the parameters that are used by the schema file, which is used, in turn, by a server. This is a special type of resource class that consists of a single record. See “Creating the resource definitions for adding queues to the SFS server” on page 59 for more information about SCD use.

Structured File Server Definitions (SSD)

The Structured File Server Definitions (SSD) define all the servers that are available to your region. See “Creating the resource definitions for adding queues to the SFS server” on page 59 for more information about SSD use.

Resource definitions that affect programs and transactions

Resource definitions that affect programs and transactions identify the application components that are accessed within and across regions to perform tasks within a CICS system.

Program Definitions (PD)

The Program Definitions (PD) define programs, map sets, and tables. You can define remote and local programs in the PD. See “Configuring Program Definitions (PD)” on page 127 for more information about PD use.

Transaction Definitions (TD)

The Transaction Definitions (TD) identify transactions. CICS uses this definition to verify incoming requests to start a transaction, and to supply information about the transaction. You can define remote and local transactions in the TD. See “Configuring Transaction Definitions (TD)” on page 130 for more information about TD use.

Comparing resource definitions across CICS family members

On the mainframe, CICS uses tables for resource storage. CICS on Open Systems uses region-related storage definitions instead. Table 3 shows the correspondence between the two:

Table 3. Resource definition types

Mainframe CICS	TXSeries for Multiplatforms
Destination Control Table (DCT)	Transient Data Definitions (TDD)
File Control Table (FCT)	File Definitions (FD)
Journal Control Table (JCT)	Journal Definitions (JD)
Monitoring Control Table (MCT)	Monitoring Definitions (MD)
Program Control Table (PCT)	Transaction Definitions (TD)
Program List Table (PLT)	These are included in the Region Definitions (RD)
Processing Program Table (PPT)	Program Definitions (PD)
System Initialization Table (SIT)	Region Definitions (RD)
Signon Table (SNT)	User Definitions (UD)
Terminal Control Table (TCT)	Terminal Definitions (WD) Communications Definitions (CD)
Temporary Storage Table (TST)	Temporary Storage Definitions (TSD)
Transaction List Table (XLT)	This is included in the Transaction Definitions (TD)

Using shorthand in the resource definitions

CICS provides the following shorthand methods for specifying the value of some CICS attributes:

- %R or %r** Expands to the current region name. For example, when a reference is made to the auxiliary temporary storage queue for a region that has a DB2 file manager, and that region is named **regionA**, the shorthand reference of **%Rreqtsq#** is automatically expanded by CICS to be **regionAreqtsq#**.
- %H or %h** Expands to the current host name. For example, when a reference **%H** is used on a machine named **HostABCD**, the **%H** is automatically expanded by CICS to **HostABCD**.
- %S or %s** Expands to the short name of the current SFS server or PPC Gateway server. For example, when a reference **%S** is made to a PPC Gateway server named **cicsgwy**, the **%S** is automatically expanded by CICS to **cicsgwy**.

CICS expands the shorthand notation wherever it occurs. If you need to express the percent sign (%) as a literal character, use two consecutive percent signs (%%).

Tools for configuring CICS regions and resources

This section describes the command utilities that are used to administer CICS regions and resources.

The types of tasks that are performed while you are configuring and administering a CICS system are the same on all platforms. Regardless of platform, a CICS system administrator must exist with the appropriate privileges to administer the CICS system and its resources. Additionally, every CICS system requires definitions for many types of resources, such as queues, files, regions, clients, and users. These definitions are created and modified by the **cicscp** control program commands, the CICS resource management commands that are called by the **cicscp** control program commands, and operating system and native database commands.

On the AIX platform. The AIX System Management Interface Tool (SMIT) commands are available and are recommended for use on the AIX platform.

The CICS control program commands

The CICS control program (**cicscp**) commands combine the basic CICS resource management commands to complete specified administration and configuration tasks. By using the **cicscp** command programs, you can ensure that the underlying CICS resource management commands are issued in the correct sequence. The **cicscp** command programs automate the common, frequently performed administration tasks.

Several **cicscp** command programs coordinate configuration tasks. The commands that are used for configuration are **cicscp create** and **cicscp destroy**. The commands that are used for operations are **cicscp start** and **cicscp stop**. Some of the objects on which you can use these options are **ppcgwy_server**, **region**, and **sfs_server**.

If no arguments are supplied to the **cicscp** command, it enters an interactive mode through which the user can enter the required information in response to prompts.

The **cicscp** command set is strongly recommended for configuring a CICS system. It does the configuration by using default values where necessary and by imposing some naming conventions. Within this framework, it also provides flexibility for customization.

You can use the **cicscp** commands to fully automate the creation of a simple configuration. However, for more complex configurations and to update existing resource definitions, you use other CICS resource-management commands.

For information about how to use the **cicscp** command to set up a region, see “Examining the effects of using the **cicscp** command to set up a region” on page 18.

For more information about the full set of options that are available with **cicscp**, see the *TXSeries for Multiplatforms Administration Reference*.

The CICS resource management commands

Although the **cicscp** commands are the recommended method for setting up CICS, the CICS resource management commands, which underlie the CICS control program commands, are helpful when you are working with a complex

configuration. In complex environments, it can be beneficial to set up each CICS component step-by-step. The resource management commands are also known as Resource Definition Online (RDO) commands. This document refers to these commands as CICS resource management commands. Appendix B, “Using CICS resource management commands for system management,” on page 269 describes how to use these commands to do specific tasks.

For a list of CICS resource management commands that are most commonly used to set up a CICS system and to prepare a region for production use, see “Commands used to set up a CICS system” on page 270 and “Commands used to prepare a region for production” on page 270.

Other commands used to administer a CICS system

Some commands that are issued during a CICS configuration are specific to the operating system. In these cases, the documented commands are platform specific. For example, CICS requires file volumes for the storage of data. The creation and maintenance of file volumes is an operating system function that is done by operating-system utilities. Therefore, when a CICS procedure requires some modification to a file volume, the procedure indicates a platform-specific command that is appropriate to the activity.

Similarly, programming languages, such as Micro Focus Server Express COBOL, relational databases, and file management systems, such as DB2 and SFS, provide commands that can be used in conjunction with a CICS system. Therefore, documentation of database or COBOL procedures that is used in CICS processes includes product-specific commands that are appropriate to the activities.

CICS Application Programming Interface (API) commands and CICS-supplied transactions are also used to administer CICS systems.

The AIX System Management Interface Tool

On the AIX platform. The System Management Interface Tool (SMIT) for CICS for AIX provides both a graphical user interface (GUI) and a text-based menu interface to many of the CICS configuration and management tasks.

The AIX GUI tool is invoked by the **smit** command, and the text-based menu tool is invoked by the **smitty** command. Both the **smit** and **smitty** tools perform the same tasks as do the **cicscp** commands and the CICS resource management commands. Use of these tools is encouraged for performing the procedures that are described in this guide. In particular, they simplify the creation and modification of CICS regions, SFS file servers, and the resources that a region uses.

Use of the AIX SMIT tool is detailed in Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259. The AIX SMIT examples that are used in this book show the text-based displays that the **smitty** command invokes.

Chapter 2. Enabling administrator, developer, and end user access to CICS

This chapter discusses how to establish the permissions required for the three types of CICS users to have access to CICS. The CICS system administrator needs to perform CICS administration tasks on the operating system and with DB2 or the Structured File Server (SFS) server. The CICS developer needs to have access to the tools and libraries that are used to build applications. The CICS user needs to have access to CICS regions and gather information from files and databases.

The configuration of environment variables that affect clients also affects the CICS system's behavior for an individual user. These environment variables are discussed in "Setting the language for the user's client connection" on page 15.

Setting up a CICS system for an administrator

Many CICS administration procedures require that you log in to the operating system with a user ID that has root privileges. Some procedures do not work unless you are actually logged in as **root**. Set up the **root** user ID for CICS system administration and use this user ID for all CICS administrative tasks.

You can enable non-root user users to perform CICS administrative tasks. This is described in Appendix C, "Performing CICS administrative functions as a non-root user," on page 285.

If you are configuring a region to use a DB2 database for CICS queue and file management, you must be set up as an administrator for the instance that the region will be using. Refer to the DB2 documentation for more information.

To set up the appropriate privilege for a CICS system administrator:

1. Verify that **root** is a member of the **cics** and **cicsterm** groups. (Information about this step is included in the *TXSeries for Multiplatforms Installation Guide*.)
2. Set the environment variables that are specified in Table 4 on page 12 for the AIX platform, Table 5 on page 12 for the HP-UX platform, or Table 6 on page 13 for the Solaris platform.

Table 4. Environment variables used for CICS administration on the AIX platform

Set or expand:	To include:
LANG	One of the following, depending on your locale: • de_DE (German) • en_US (U.S. English) • es_ES (Spanish) • fr_FR (French) • it_IT (Italian) • ja_JP.eucjp (Japanese MBCS EUC) • Ja_JP.IBM932 (Japanese DBCS PC) • ko_KR (Korean) • pt_BR (Brazilian Portuguese) • zh_CN (Simplified Chinese) • Zh_CN (Simplified Chinese GB18030) • zh_TW (Traditional Chinese) Note: zh_CN is the short code for zh_CN.eucCN, Zh_CN is the short code for Zh_CN.GB18030.
PATH	/usr/lpp/cics/bin
LIBPATH	/usr/lpp/cics/lib
NLSPATH	/usr/lpp/cics/msg/%L/%N
CICSREGION	The name of the region

Table 5. Environment variables used for CICS administration on the HP-UX platform

Set or expand:	To include:
LANG	The appropriate value for your language as follows (if this variable is not set, it defaults to C): • de_DE.iso88591 (German) • en_US.iso88591 (U.S. English) • es_ES.iso88591 (Spanish) • fr_FR.iso88591 (French) • it_IT.iso88591 (Italian) • pt_PT.iso88591 (Portuguese) • ja_JP.eucJP (Japanese) • ja_JP.SJIS (Japanese) • ko_KR.eucKR (Korean) • zh_CN.hp15CN (Simplified Chinese) • zh_CN.gb18030 (Simplified Chinese GB18030)
PATH	/opt/cics/bin
SHLIB_PATH	/opt/cics/lib
NLSPATH	<i>install_dir</i> / msg/%L/%N /opt/cics/msg/%L/%N
CICSREGION	The name of the region

Table 6. Environment variables used for CICS administration on the Solaris platform

Set or expand:	To include:
LANG	The appropriate value for your language as follows. (If this variable is not set, its value defaults to C, which corresponds to U.S. English): • de (German) • en_US (U.S. English) • es (Spanish) • fr (French) • it (Italian) • ja (EUC Japanese) • ja_JP.PCK (Japanese, PC code page) • ko (Korean) • pt_BR (Brazilian Portuguese) • zh (Simplified Chinese) • zh_CN.GB18030 (Simplified Chinese GB18030) • zh_TW.BIG5 (Traditional Chinese)
PATH	/usr/ucb /opt/cics/bin
LD_LIBRARY_PATH	/opt/cics/lib
NLSPATH	/opt/cics/msg/%L/%N
CICSREGION	The name of the region

Deciding where to set your environment variables

Base your decision about where to set these environment variables on which users you want to have access to these settings. If you do not want every user to have access to these environment variables settings, you can set them in **root**'s profile and in the individual user profiles. Or, because all CICS users require these values, you can set them in the directory location that your operating system uses for setting environment variables. Table 7 lists the directory locations that each platform uses for setting environment variables.

Table 7. Where to set environment variables on the AIX, HP-UX, and Solaris platforms

Platform	Directory location
AIX	/etc/environment
HP-UX	/etc/profile
Solaris	/etc/profile

For more information about environment variables, see the *TXSeries for Multiplatforms Administration Reference*.

Considering setup requirements for CICS developers

The CICS developer falls into a special class of users. The developer needs to build programs that process transactions and modify files that exist in CICS regions and databases. An administrator needs to consider the precise access permissions that a developer needs.

The administrator needs to ensure that access to all the required development tools and libraries is included in the developer's PATH environment variable. In addition, the administrator needs to consider the permission levels that are required for the developer both in test and in production environments.

To access the CICS commands that are needed for development, the developer's user account does not need to be included in either the **cics** or the **cicsterm** group. However, access to executable commands and libraries that are used in development must be provided in the developer's PATH and library path environment settings.

Note: User accounts for developers who require access to the following CICS commands must be included in the **cics** group:

- **cicsddt**, used for work with DB2 files
- **cicstfmt**, used to format traces
- **cicsmfmt**, used to format monitoring output
- **cicsdfmt**, used to format dumps

At a minimum, the developer's PATH setting must include the **\$CICS/bin** directory, and the operating system's library path environment setting must include **\$CICS/lib**. The **cicstcl** command, which is used to compile CICS applications, accesses these libraries.

If the developer is using external products, such as COBOL, WebSphere MQ®, DB2, or other relational database products, the environment settings for the developer must include access to the directories that those products use for their commands and libraries. For example, if a developer is using COBOL, ensure that the developer's PATH setting includes the **\$COBDIR/bin** directory and the developer's operating system library path includes the **\$COBDIR/lib** directory (where **\$COBDIR** expands to the COBOL product's home directory).

Additionally, some products require that the developer's user account is included in the product's administration group. Follow the instructions that are provided by the external product to ensure the appropriate access.

Setting up a CICS system for an end user

When a user is given access to a CICS system, the system administrator provides for the appropriate access levels and the appropriate security levels. This section discusses the problems that are addressed by the system administrator when access is enabled to a CICS system.

Adding a new user to the region

A CICS user connects to a CICS region by using one of the clients that are described in Chapter 6, "Working with CICS clients," on page 71. If a user does not have a User Definitions (UD) entry, the user connects to the region with the region's default user ID (see "Creating the region's default user ID" on page 272).

The various attributes that are set for users in the User Definitions are described in the *TXSeries for Multiplatforms Administration Reference*. Of particular interest are the following:

- **RSLKeyList** contains the list of resource security keys for this user.
- **TSLKeyList** contains the list of transaction security keys for this user.

- **CICSPassword** contains a password that is used for password checking when the region is configured to use internal CICS security. (The RD **AuthenticationService** attribute is set to **CICS**.) This password is encrypted by CICS facilities; for example, the **cicsadd** command encrypts the user's password.

A CICS user must be included in the group **cicsterm**, but do not include the user in the group **cics**. Members of the group **cics** have access to the unencrypted passwords that are contained in the region's Product Definitions (XAD) files that the system's resource managers use.

The user's setting for the PATH environment variable must include the **\$CICS/bin** directory. The setting for the user's NLSPATH environment variable must include the following string:

```
install_dir/msg/%L/%N:$CICS/msg/C/%N
```

where *install_dir* expands to the CICS product's directory, and **\$CICS** expands to the CICS product directory.

Using CICS facilities to change a password

If you have CICS system administration authority, you can use the **cicsupdate** command to change the value for the **CICSPassword** attribute. In this case, the old password is not required. For example, the following command sets the **CICSPassword** attribute for the user **SALLY** to an encrypted version of **ORZ87TU**:

```
cicsupdate -c ud -r regionA -B SALLY CICSPassword=ORZ87TU
```

In addition, users can change their own passwords by using the CICS-supplied transaction CESN. See the *TXSeries for Multiplatforms Administration Reference* for more information about using CESN.

Setting the language for the user's client connection

Use this procedure to specify the language that you want to use for issuing messages to the user when a client connection is made to a CICS region.

Messages to the client are issued from the region and from the client; therefore, you must do the following:

- Verify that the appropriate message catalogs are installed on each host with a client. For example, if one user of the client requires English and another user of the client requires Japanese, verify that English and Japanese message catalogs are both installed. Table 8 shows the location of the message catalogs for each host with a client.

Table 8. Location of the client message catalogs

Platform	Directory
AIX	/usr/lpp/cics/msg
HP-UX	/opt/cics/msg
Solaris	/opt/cics/msg

Note: On HP-UX, Portuguese Brazilian messages are issued through the Portuguese locale (pt_PT.iso88591).

- Verify that the region is set up correctly so that it can send messages to the client in the language that the client requires. See “Setting the language for the region” on page 21.

To specify the language to display to the user through the client connection:

1. Add the NLSPATH path in the directory that is appropriate to your platform, as indicated in Table 9.

Table 9. NLSPATH environment variable settings

Directory	Locale example
AIX: /etc/environment	<i>install_dir/msg/%L/%N:/usr/lib/nls/msg/en_US/N</i>
HP-UX: /etc/profile	<i>install_dir/msg/%L/%N:/usr/lib/nls/msg/En_US/N</i>
Solaris: /etc/profile	<i>install_dir/msg/%L/%N:/usr/lib/nls/msg/C/N</i>

2. Set the LANG environment variable in the end user’s environment to the language that is required for that user.

Table 4 on page 12, Table 5 on page 12, and Table 6 on page 13 show the possible LANG variable settings and locale settings for each platform.

See “Program Definitions and setting up Basic Mapping Support for different languages” on page 128 for information about setting up the Program Definitions (PD) for different languages.

Chapter 3. Configuring and modifying CICS regions

The *TXSeries for Multiplatforms Installation Guide* describes how to create a default CICS region. This chapter assumes that you have created the default region.

In this chapter:

- “Creating a region by using the CICS control program command,” “Examining the effects of using the `cicscp` command to set up a region” on page 18, and “Deleting a region by using the `cicscp` command” on page 19 discuss the behavior that is invoked when the **cicscp** command is used for creating, starting, and deleting a region that is using the **cicscp** command programs.
- “Creating a region by using the CICS resource management commands” on page 20 indicates how and when to use the CICS resource management commands to create a region.
- “Changing the Region Definitions (RD) attributes of a new region” on page 20 discusses how to customize the default region that is created by using the **cicscp** command programs.
- “Setting the language for the region” on page 21 discusses setting up a language for use by the CICS region that is different from the language that is used on the host machine.
- “Setting up a separate file system for the CICS log directory” on page 22 discusses setting up a separate file system to hold dynamic log data for a configured CICS region.

Creating a region by using the CICS control program command

The **cicscp** command set is recommended for the creation of CICS regions.

The *TXSeries for Multiplatforms Installation Guide* provides detailed instructions about how to create and start an SFS server or a DB2 database before a region is created. It also describes how to create a default region by using the **cicscp create region** command, which automatically creates a default SFS server if one does not already exist.

The **cicscp create region** command with appropriate options is used to create a configuration with standard default names and values.

Figure 1 on page 18 gives a high-level overview of how to use the **cicscp create region** command that is described in the *TXSeries for Multiplatforms Installation Guide*.

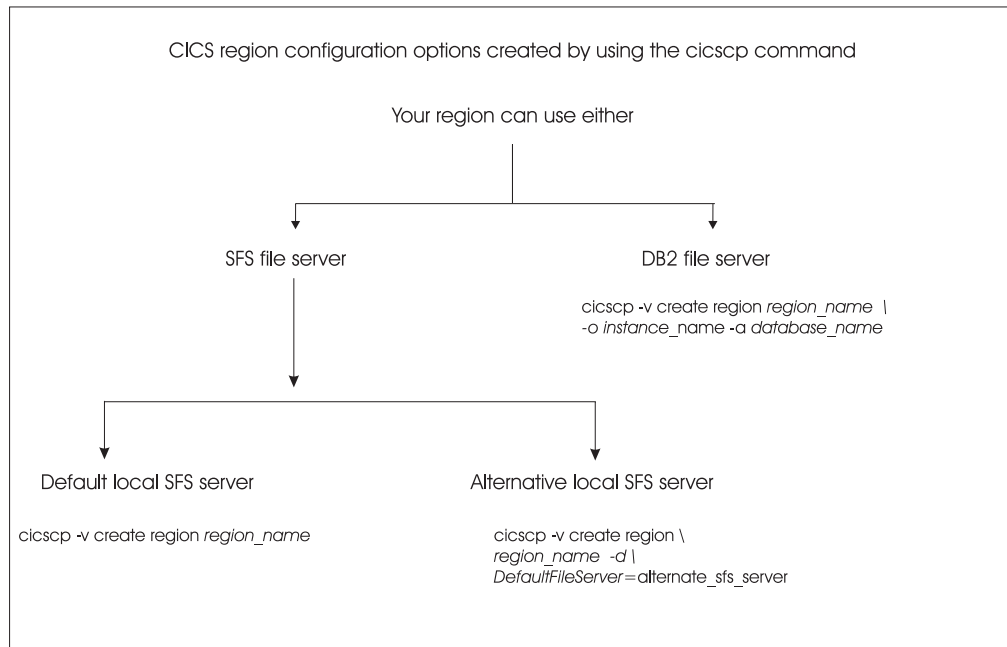


Figure 1. Region configuration options set up by using the cicscp command

The specific details that result from the use of the commands that are shown in Figure 1 are discussed in “Examining the effects of using the cicscp command to set up a region” and “The cicscp defaults.”

Examining the effects of using the cicscp command to set up a region

To examine the effects of using the **cicscp create region** command, consider the results when the following command is used to create a region named **regionA**, using the default definitions:

```
cicscp -v create region regionA
```

This command configures all necessary components to create the region, specifically, the SFS server, and the region itself.

This command also creates the necessary operating system user accounts and the internal files that the region uses. The processing of the **cicscp** command is based on the information that is contained in the resource definitions. It does not, however, set up the SFS server if it is already configured.

If the **cicscp** command detects that the existing configuration of the SFS server is invalid, it issues a message that suggests that the configuration is to be destroyed and reconfigured.

The cicscp defaults

The **cicscp** command uses the following defaults:

- The user ID that is created for the region is **cics**.
- The default value for the Region Definitions (RD) **DefaultUserId** attribute is **CICSUSER**.
- The SFS server is named after the host name. For example, if the host name is **hostABCD**, the SFS server is named: **././cics/sfs/hostABCD**.

- A short name is created for the SFS server; it consists of the letter **S** followed by the first seven characters of the host name. For example, if the host name is **hostABCD**, the short name is **ShostABC**.
- The default SFS user ID is named by using the short name.
- SFS logical volume names are based on the short name. For example, if the short name is **ShostABC**, the logical volume that is used for queue and file data is named **sfs_ShostABC**, and the logical volume that is used for the queue and file log is named **log_ShostABC**.

Note: When a region is created on any platform by using the CICS resource management command **cicsdefault**, or when the SMIT tool is used to create a region on the AIX platform, the values that are used as the default names for the SFS logical volumes are different from the default values that are imposed when the **cicscp** command is used for this task. The default values that are imposed with the **cicsdefault** command or with the use of the SMIT tool are **log_SFS_SERV** and **sfs_SFS_SERV**.

- **On the AIX and HP-UX platforms.** When the **cicscp** command is used on the AIX or HP-UX platform, it creates an SFS logical volume that has the default size of 64 MB.

The **cicscp** command has more options than those that are shown in this document. You can use additional options to import a set of your own Region Definitions (RD). You can use these definitions instead of using the default definitions and specifying a group ID that is to be used for the File Definitions (FD). If you are using DB2 for file management, you can specify DB2 explicit authentication or single-phase commit.

For more information about the other options that are available with **cicscp**, see the *TXSeries for Multiplatforms Administration Reference*.

If you do not want to use the defaults that the **cicscp** command imposes, you can set environment variables to override the command defaults before you use the **cicscp** command. See “Overriding the defaults imposed by the **cicscp** start **sfs_server** command” on page 29 for more information about this topic. If you do not want **cicscp** to create logical volumes of the default size, create the logical volumes before you use the **cicscp** command. Follow the procedure that is described in “Preparing the physical storage devices for the SFS server” on page 274.

Deleting a region by using the **cicscp** command

Use this procedure to delete a region by using the **cicscp** command. This procedure does not remove the operating system user ID that was created for the region.

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the command:

```
cicscp -v destroy region regionName -s
```

This command:

- Stops the region if it is running
- Deletes the region definitions from the CICS region database

- Deletes the entry in the operating system mechanism that is used for starting and stopping long-running processes
- Removes the region directory `/var/cics_regions/regionName`
- When the `-s` option is included, deletes the resource definitions that associate Transaction Definitions (TD) and Temporary Storage Definitions (TSD) in the file server's queues

See the *TXSeries for Multiplatforms Administration Reference* for more information about using this command.

For information on using the CICS resource management commands to delete a region, see “Using the `cicsdestroy` command to delete a region” on page 282.

Creating a region by using the CICS resource management commands

The CICS resource management commands can be used for creating regions when a region other than the default region that is invoked by the `cicscp` command set is required.

Multiple servers and regions can be created on the same host, and require naming schemes other than the defaults that the `cicscp` command imposes. Although the `cicscp` commands supply options that enable nondefault names to be used, you can use the CICS resource management commands instead for speed and problem resolution. Because the CICS resource management commands do not perform all the prerequisite checking that is done by the `cicscp` command, they complete more quickly. In the case of a command failure, the immediate success or failure of the resource management command makes it easier to identify where the problem is.

CICS provides resource management commands that enable the creation of a default region or the duplication and re-creation of a customized region.

See Table 39 on page 271 for information on how to combine the CICS resource management commands in the following conditions:

- Setting up a region that uses a local SFS server for CICS file and queue management
- Setting up a region that uses a remote SFS server for CICS file and queue management
- Setting up a region that uses a DB2 server for CICS file and queue management

Changing the Region Definitions (RD) attributes of a new region

This section discusses how to make permanent changes to the Region Definitions (RD) for a region that has just been set up but has not yet been put into production. The first time a region is started is always a cold start. A cold start is required so that any changes that have been made to attribute values in the permanent database can take effect.

Attention: Perform this procedure before you begin to use the region in a production environment. A cold start is required for the new attribute values to take effect.

Before using this procedure, review the information about Region Definitions that is given in the *TXSeries for Multiplatforms Administration Reference*. Do not change any of the RD file server attributes unless you intend to reconfigure the file manager.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.

2. Shut down the region. Enter the command:

```
cicscp -v stop region regionName
```

3. Make the changes to the RD attributes by using the **cicsupdate** command. For example:

```
cicsupdate -c rd -r regionA MaxServer=10 MinServer=5
```

In this example, the **cicsupdate** command modifies the Region Definitions (RD) attributes **MaxServer** and **MinServer**. The updates to these attributes modify the limits for the maximum and minimum number of concurrently running application servers.

4. Start the region. Enter the command:

```
cicscp -v start region regionName StartType=cold
```

Note: If this is the first time that the region has been started, a cold start is automatically used. If the region has been started before, a cold start must be specified by adding the attribute setting **StartType=cold** to the **cicscp start region** command.

The default values for these attributes in the RD stanza are set automatically when the **cicscp** commands are used to create the region. The **cicsupdate** command allows the defaults to be adjusted to meet the business needs better. (More information about the **cicsupdate** command and about the RD attributes is available in the *TXSeries for Multiplatforms Administration Reference*.)

Setting the language for the region

You can specify that the region is to use a different language from the language that the host machine uses. Before you specify a different language, ensure that:

- The region is configured.
- The appropriate message catalogs are installed. For example, if one client requires Chinese, another client requires Japanese, and the region requires English, verify that Chinese, Japanese, and English are installed. Table 4 on page 12, Table 5 on page 12, and Table 6 on page 13 show the available message catalogs for each platform.

To specify the language that the region is to use when it issues messages:

1. Add the information for the NLSPATH variable that is shown in Table 4 on page 12, Table 5 on page 12, or Table 6 on page 13, depending on your platform.
2. Set the LANG environment variable in the file **/var/cics_regions/*regionName*/environment** to the language that is required for the region. Table 4 on page 12, Table 5 on page 12, and Table 6 on page 13 show the possible LANG variable settings for each platform.

Note: On HP-UX, Portuguese Brazilian messages are issued through the Portuguese locale (pt_PT.iso88591).

Setting up a separate file system for the CICS log directory

The CICS log directory, `/var/cics_regions/regionName/log`, holds the dynamic log data for the CICS region. The product installation automatically installs the log directory into the system's `/var` volume. If the log entries fill the `/var` volume, system performance can be affected. A separate file system for the CICS log can be used, but it cannot be set up until after the region is configured.

Note: Because the information contained in the CICS log is required to warm start the region, it is recommended that you mirror the log so that you can recover the system to the point of failure if data is lost. For more information about the importance of the CICS log and recovery, see Chapter 20, "Understanding CICS backup and recovery," on page 239.

Chapter 4. Working with a file manager in a CICS region

The CICS file management system consists of the file manager, which is a basic file structure, queues, and the user's data files. In addition, CICS can use journal files, extrapartition queues, and files that are associated with relational database management systems (RDMS). These files reside in the operating system file structure. This complete structure is created in stages while the CICS system is configured.

This chapter discusses the creation of the file manager. The file manager must be configured and started before any CICS region that is to use it is started. The file manager oversees the administration of files that hold the user's data and queues that pass data and information about the state of the CICS system. CICS uses either a Structured File Server (SFS) server or a DB2 or Oracle database as the file manager.

For information about what you must consider when you choose the file manager, see *TXSeries for Multiplatforms Concepts and Planning*.

In this chapter:

- "Administration of file and queue management on an SFS server" discusses the use of SFS to perform the file management tasks.
- "Administration of file and queue management on DB2" on page 31 discusses the use of DB2 to perform the file management tasks.
- "Administration of file and queue management on Oracle" on page 40 discusses the use of Oracle to perform the file management tasks.
- "Migrating file and queue management (DB2 and SFS)" on page 49 explains the process of moving between the DB2 and SFS file management options.
- "Migrating file and queue management (Oracle and SFS)" on page 53 explains the process of moving between the Oracle and SFS file management options.

Administration of file and queue management on an SFS server

When you install CICS, an SFS file manager is installed automatically. To configure this SFS server, follow the instructions given in the *TXSeries for Multiplatforms Installation Guide*. (You will have already done this if you have run the Installation Verification Program.)

"How a CICS system uses an SFS file manager" on page 24 gives an overview of an example CICS system that is using an SFS file manager. This overview shows how the CICS system uses the file manager, intrapartition queues, application data files, journal files, extrapartition queues, and files that are associated with a RDMS.

If you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide* to set up a default CICS system, see also "Meeting file manager requirements by using `cicscp` commands" on page 27, which describes the state of your system. The section also describes how to reach the same configuration state for your SFS server by using the CICS resource management commands instead of the `cicscp` command set.

“Overriding the defaults imposed by the `cicscp start sfs_server` command” on page 29 describes how you can change the standard names and sizes that are supplied by the **cicscp** command programs, to customize your CICS implementation.

This chapter also discusses the following SFS administration topics:

- “Enlarging the logical volumes for the SFS server” on page 30
- “Using `nlsString` data types in SFS files” on page 30
- “Deleting an SFS server by using the `cicscp` command” on page 31

How a CICS system uses an SFS file manager

The *TXSeries for Multiplatforms Installation Guide* describes how you can use the **cicscp create sfs_server** and the **cicscp start sfs_server** command programs to create an SFS file server. If you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, you have created a default file manager to manage the region’s queues and the user’s data files.

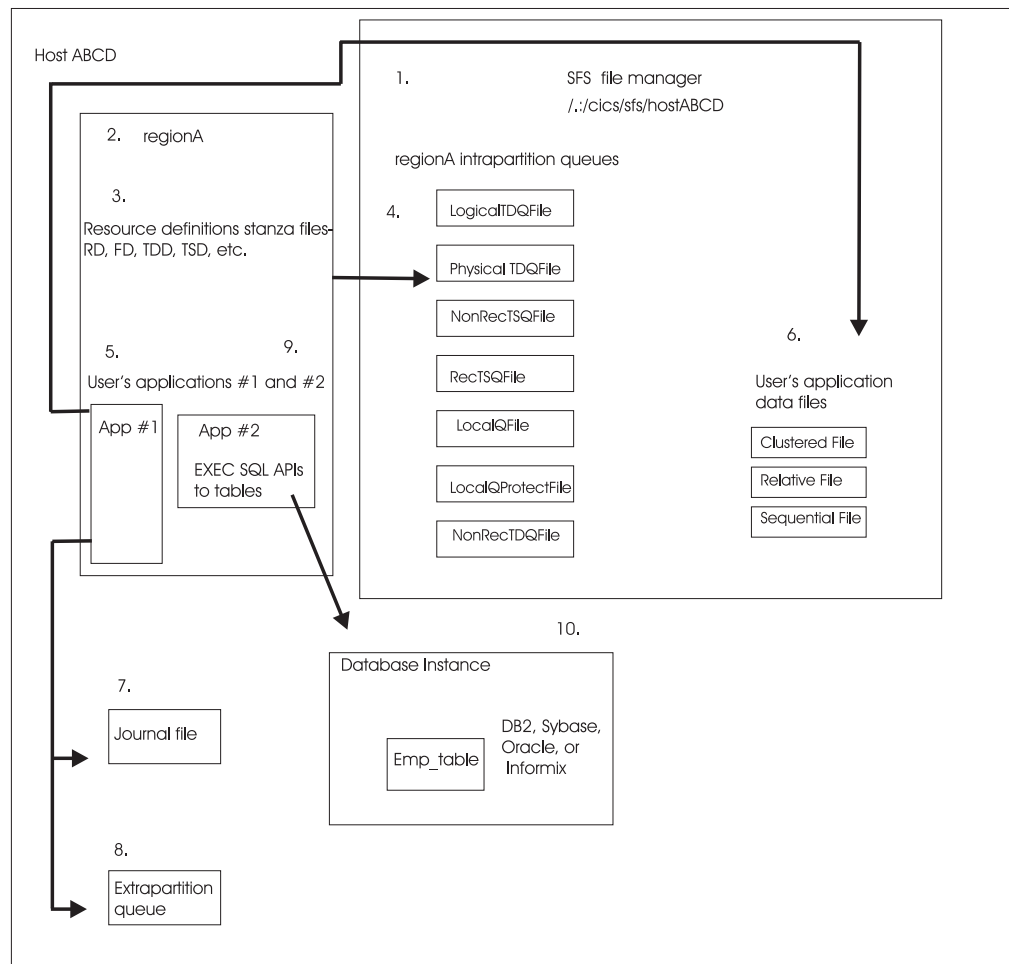


Figure 2. An example of an SFS file management system for CICS

Figure 2 shows an example of CICS file management system that uses an SFS file manager. In this example, the base configuration (a region, a file manager, and the intrapartition queues) can be built by using either the **cicscp** command programs or CICS resource management commands. Access to the user’s application data files, the extrapartition queue, the journal file, and the database tables is enabled

through resource definitions that are added to the region's resource definition stanza files. The resource definitions that are required to identify these file structures to CICS are the same, regardless of what commands are used to create the CICS system with which they interact.

Figure 2 on page 24 shows an example of how a CICS system uses files and queues. The example CICS system resides on host machine hostABCD. An SFS server is acting as the file manager for this system. This SFS file manager is managing files and queues for regionA. In this example, the user has created an application, App#1, on regionA that stores its data in files that are managed by the SFS file manager. This application also makes use of a journal file and an extrapartition queue, neither of which are managed by the SFS file manager. Instead, these files are managed by the operating system file structure.

In addition, on regionA, the user is using a relational database application, App#2, to access data that is stored in tables. The application can be on any of the relational database products that are supported by CICS: DB2, Oracle, Sybase, or Informix®. The SFS file manager does not manage the relational database tables. That job is done by the database product itself.

This section uses the numbered areas that are shown in Figure 2 on page 24 to discuss the ways in which a CICS system, created either by the **cicscp** commands or by the CICS resource management commands, can use files and queues. The pertinent CICS commands and resource definitions that are related to each numbered area are described.

1. The SFS file manager is shown in Area 1.

- The file manager must exist and must be running before the region that is to use it is created. The file manager can be created through a CICS control program (**cicscp**) command. If the user follows the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, and uses the **cicscp** commands, all the defaults that accompany the **cicscp** command programs apply to this diagram. By default, the SFS server takes the first seven characters of the name of the host machine. So when the user issues the command **cicscp create sfs_server** on the machine hostABCD, an SFS file manager is created and is located in the directory `./cics/sfs/hostABCD`. The **cicscp start sfs_server** command activates the SFS file manager.
- Alternatively, the user can create a file manager to manage files and queues on regionA by using the CICS resource management command **cicssfscree**. After the use of the **cicssfscree** command to start the file manager, the region that is to use it can be created.

2. When the file manager is running, the region that is to use it can be created.

Area 2 in Figure 2 on page 24 shows a region named regionA.

- If the user follows the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, the command **cicscp create region regionA** is issued. When the user issues this command, the resource definition stanza files, which are shown in Area 3, and the intrapartition queues, which are shown in Area 4, are automatically created. The **cicscp start region regionA** command activates the region and connects it to the SFS file manager.
- Alternatively, the user can create this region by using the command **cicsdefault**. This command is a CICS resource management command that creates a default region by using the default definitions that are supplied by CICS.

3. Area 3 in Figure 2 on page 24 shows the file structure that is created to hold the resource definitions that define all resources that regionA uses.
 - The resource definition file structure is automatically created when the **cicscp create region** command is issued. The user adds a resource definition entry to identify each resource that is available to the CICS system. (Chapter 8, “Working with CICS resource definitions,” on page 115 identifies the classes of resource definitions and describes the CICS resource management commands that are used to create these definitions.)
 - Alternatively, the user can create the resource definition file structure by using the **cicsdefaultserver** command. The user then adds a resource definition entry to identify each resource that is available to the CICS system.
4. Area 4 shows the intrapartition queues that the CICS system uses.
 - These queues are automatically created by the **cicscp create region** command. Of these seven queues, only the **NonRecTDQFile** queue can be accessed without the addition of a resource definition entry to identify it to the CICS system. The **LogicalTDQFile**, the **PhysicalTDQFile**, the **LocalQFile**, and the **LocalQProtectFile** queues must be named in a Transient Data Definition (TDD) stanza file. The **NonRecTSQFile** and the **RecTSQFile** queues must be declared in a Temporary Storage Definition (TSD) stanza file. After the TDD or TSD entry is made, CICS can use the CICS application programming interface (API) commands to access these queues.
 - Alternatively, the user can create these intrapartition queues by using the **cicssfscnf** command. Again, TDD and TSD entries are needed to identify the queues to CICS; and similarly, only the **NonRecTDQFile** queue can be accessed without a resource definition being added to identify it to the CICS system.
5. Area 5 shows an application, App#1.
 - App#1 uses EXEC CICS APIs to access files that are managed by the SFS file manager. The application is identified to CICS through a Program Definition (PD) entry in the PD stanza file.
 - App#1 uses the same file access methods and is identified to CICS through the same Program Definition (PD) entry, regardless of which commands were used to create the CICS system in which the application is running.
6. Area 6 shows the files that hold the data for the user’s application, App#1.
 - These files, which can be any of three types: clustered, relative, or sequential, are designed to simulate Virtual Storage Access Method (VSAM) files that are used on CICS mainframe systems. The application’s data files are defined to CICS through a File Definitions (FD) entry and accessed through EXEC CICS API commands in the application.
 - Regardless of which commands were used to create the CICS system in which the application is running, the types and files that can be used, the definitions that are needed to identify them to CICS, and the commands that are used to access the files are the same.
7. Area 7 shows a journal file that is used by App#1, which is identified in Area 5.
 - A journal file is not managed by the file manager. It is an ASCII file that is managed by the operating system’s file structure. It is identified to CICS by a Journal Definitions (JD) entry and accessed through EXEC CICS API commands that are embedded in the application.

- A CICS application uses a journal file in the same way, regardless of which commands were used to create the CICS system in which the application resides.
8. Area 8 shows an extrapartition queue that is used by App#1, which is identified in Area 5.
 - An extrapartition queue is not managed by the file manager. It is an ASCII file that is managed by the operating system's file structure. It is identified to CICS by a TDD entry and accessed through EXEC CICS API commands that are embedded in the application.
 - A CICS application uses an extrapartition queue in the same way, regardless of which commands were used to create the CICS system in which the application resides.
 9. Area 9 shows a relational database application, created by the user, that resides on regionA.
 - This application uses EXEC SQL commands to access database tables that reside in the operating system's file structure. The application is identified to CICS through a Program Definition (PD) entry in regionA's PD stanza file.
 - A relational database application uses tables in the same way, regardless of which commands were used to create the CICS system in which the application resides.
 10. Area 10 shows the database tables that hold the data for the application identified in Area 9.
 - These tables are managed entirely by the database product. FD entries are needed to identify these tables to CICS.
 - A relational database application makes use of tables in the same way, regardless of which commands were used to create the CICS system in which the application resides.

Meeting file manager requirements by using **cicscp** commands

In the configuration that is shown in Figure 2 on page 24, the **cicscp create sfs_server** command that is issued in Area 1 manages several conditions that enable the file manager to be used with CICS:

- An operating system user ID for the SFS server exists.
- Two logical volumes are created, one for CICS queues and files, and one for logging information about the CICS queues and files.

Note:

- **On the AIX and HP-UX platforms.** The **cicscp** commands create the logical volumes.
- **On the Solaris platform.** You must create these volumes by using Solaris operating system commands before you use the **cicscp create sfs_server** command. See the *TXSeries for Multiplatforms Installation Guide* or your Solaris operating system documentation for information about completing this step.
- The **/var/cics_servers/SSD** and **/var/cics_servers/SCD** directories exist. The SSD directory contains the following:
 - A lock file that is used to lock access to the SFS server configuration in the SSD directory while the SFS server is running or its configuration is being changed. The full path name for the lock file is **/var/cics_servers/SSD/cics/sfs/serverId/lock**. CICS creates this file as needed.

- A file that contains the messages that are logged for the server. CICS creates this file as needed. The full path name for the messages file is **/var/cics_servers/SSD/cics/sfs/serverId/msg**.
The SFS server creates this file as needed.
- Two restart files that contain information that the SFS server needs for a warm start. The full path names for the restart files are **/var/cics_servers/SSD/cics/sfs/serverId/restart** and **/var/cics_servers/SSD/cics/sfs/serverId/restart.bak**.
The SFS server creates these files as needed.

You can use CICS resource management commands to bring your SFS server to this same configuration state in discrete steps. See “Meeting requirements for a file manager by using CICS resource management commands.”

The addition of the queues to the file manager that is shown in Area 4 of Figure 2 on page 24 is discussed in Chapter 5, “Working with CICS queues,” on page 59. The addition of a user’s application data files that are shown in Area 6 is discussed in Chapter 10, “Working with application data files in CICS,” on page 133.

Meeting requirements for a file manager by using CICS resource management commands

The following list relates the CICS resource management commands that create the SFS file server, to the requirements that are identified in “Meeting file manager requirements by using cicscp commands” on page 27:

- Two logical volumes, one for CICS queues and files and one for logging information about the CICS queues and files, can be created by using operating system commands.

Note:

- **On the AIX platform.** The logical volumes can be created by using the graphical user interface tool, System Management Interface Tool (SMIT). See your AIX operating system documentation.
- **On the HP-UX platform.** The logical volumes can be created by using the graphical user interface tool, System Administration Manager (SAM). See your HP-UX operating system documentation.
- **On the Solaris platform.** You must create these volumes, using Solaris operating system commands, before you create the SFS server. See your Solaris operating system documentation.
- You can use CICS resource management commands to create the operating system user ID for the SFS server. See “Creating the user ID for the SFS server” on page 273 for information on setting up this ID.
- Security considerations can be met:
- When CICS resource management commands are used, the **cicsdefaultdervers** command creates the file server directory structure.
- The **/var/cics_servers/SSD** and **/var/cics_servers/SCD** directories and their required files can be created by following the procedures that are given in “Defining an SFS server by using the cicssfscree command” on page 275. In Figure 2 on page 24, this is shown in Area 4 through the use of the **cicssfscree** command.
- When the **cicssfscree** command is issued, the logical volumes are initialized and exist in an empty state.

- The addition of the region's queues to the file management system is made when the **cicssfscnf** command is issued. This topic is discussed in Chapter 5, "Working with CICS queues," on page 59.
- Finally, the application's data files are identified by resource definitions that are added to the region's resource definitions stanza files. This topic is discussed in Chapter 10, "Working with application data files in CICS," on page 133.

Overriding the defaults imposed by the **cicscp start sfs_server** command

The following procedure describes how to override the defaults that are set up when the **cicscp start sfs_server** command is used to create an SFS server:

Note: Ensure that you know the base name that is to be used in the SFS server name. For example, if the name of the host machine is **hostABCD**, the SFS server name is **./:cics/sfs/hostABCD**. The default base name of the SFS server is the host name.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.
2. **On the Solaris platform.** Ensure that your logical volumes have been created. To do this, use the Solaris operating system commands before you use the **cicscp start sfs_server** command. Follow "Preparing the physical storage devices for the SFS server" on page 274.

Note: On the AIX and HP-UX platforms, the **cicscp** commands automatically create the logical volumes.

3. If you do not want to use the defaults that the **cicscp** command uses to configure logical volumes for the SFS server, set the following environment variables:
 - **CICS_SFS_SIZE:** Specifies the size (in megabytes) both of the data and of the log logical volumes for the SFS server. The **cicscp** default is **64** each.
 - **CICS_SFS_DATA_SIZE:** Specifies the size (in megabytes) of the data logical volume for the SFS server. This overrides the environment variable **CICS_SFS_SIZE**. The **cicscp** default is **64**.
 - **CICS_SFS_LOG_SIZE:** Specifies the size (in megabytes) of the log logical volume for the SFS server. This overrides the environment variable **CICS_SFS_SIZE**. The **cicscp** default is **64**.
 - **CICS_SFS_VG:** Specifies the logical volume group both for the data and for the log logical volumes for the SFS server. The **cicscp** default is **rootvg**.
 - **CICS_SFS_DATA_VG:** Specifies the logical volume group for the data logical volume for the SFS server. This overrides the environment variable **CICS_SFS_VG**. The **cicscp** default is **rootvg**.
 - **CICS_SFS_LOG_VG:** Specifies the logical volume group for the log logical volume for the SFS server. This overrides the environment variable **CICS_SFS_VG**. The **cicscp** default is **rootvg**.
4. Use the following command to start the SFS server:


```
cicscp -v -l /tmp/cicscp.log start sfs_server ./:cics/sfs/serverId
```

The **cicscp start sfs_server** command performs the following configurations for an SFS server:

- It creates the SFS server user ID. See "Creating the user ID for the SFS server" on page 273.

- It looks for the SFS server logical volumes, and if it does not find them, it creates them. (If you have already created your logical volumes, the command finds and uses them.)
- It creates the Structured File Server Definitions (SSD).
- It starts the SFS server.

Enlarging the logical volumes for the SFS server

If a logical volume fills all the available space in the operating system file, to enlarge the logical volume, you must create a new operating system file and define it as part of the logical volume.

Two types of logical volumes are possible:

- A logical volume that is used for logging; for example, **log_Smyserv**. The name of this volume is held in the **LogVolume** attribute of the SFS server definition entry.
- A logical volume that is used for holding data; for example, **sfs_Smyserv**. The name of this volume is held in the **DataVolume** attribute of the SFS server definition entry.

The information about how to enlarge a logical volume is described in *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

Note: If you enlarge logical volumes and subsequently cold start an SFS server, any modifications that you have made to the those logical volumes are lost, and the SFS server is started with only the original logging and data volumes defined.

Using nlsString data types in SFS files

If you plan to use **nlsString** data types in SFS files, the collating language field must be set to the appropriate language value (for example, **ja_JP**) before you start the SFS server. This value is set from the **CollatingLanguage** attribute in the Structured File Server Definitions (SSD) when the SFS server is created. You can change the value by using the **cicssfsimport** command to import it, as described in the *TXSeries for Multiplatforms Administration Reference*.

After the server has been cold started, you cannot change this value without cold starting the server again.

On the AIX platform. Alternatively, you can use the fast path command **smitty cicscstartssd** to display the panel that contains the **Collating language** field. See the *TXSeries for Multiplatforms Installation Guide* for more information.

Figure 3 on page 31 shows a SMIT screen displaying the attributes for a cold start of an SFS server. The **CollatingLanguage** attribute is shown on this screen.

Cold Start an SFS Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

	[EntryFields]		
* SFS Server Identifier	././cics/sfs/sfsname		
Resource description	SFS Server Definition		
Short name used for SRC	[SFS_SERV]		
Your_Operating_System user ID for server	%S		
Buffer pool size in Kbytes	[1000]	#	
Collating language	[ja_JP]		
Idle timeout	[300]	#	
Number of Log writes per checkpoint interval	[5000]	#	
Protectionlevel	none	+	
Number of threads devoted to normal operations	[12]	#	
Number of threads devoted to resource operations	[3]	#	
First Volume (for cold start)	[sfs_%S]		

F1=Help
F5=Reset
F9=Shell

F2=Refresh
F6=Command
F10=Exit

F3=Cancel
F7=Edit
Enter=Do

F4=List
F8=Image

Figure 3. Setting NLS data types for SFS files

Note: For information on NLS settings for use with Sybase on the HP-UX platform, see “NLS issues for Sybase with CICS for the HP-UX platform” on page 173.

Deleting an SFS server by using the cicscp command

Use this procedure to delete an SFS server by using the **cicscp** command. This procedure does not remove the operating system user ID that was created for the SFS server.

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.

2. Use the following command to delete the server:

```
cicscp -v destroy sfs_server ././cics/sfs/serverId
```

This command:

- Stops the SFS server if it is running
- Deletes the SFS server definitions from the CICS server database
- Deletes the entry in the operating system mechanism used for starting and stopping long-running processes
- Removes the SFS server directory `/var/cics_servers/SSD/cics/sfs/serverId`

Administration of file and queue management on DB2

This section discusses the following topics:

- “How a CICS system uses a DB2 file manager” on page 32 discusses the creation of an example region that uses a DB2 database as its file manager.
- “Configuring a DB2 database” on page 34 describes how to configure DB2 as the file manager for a CICS region.
- “Preparing for DB2 file and queue management” on page 37 describes the creation of the database and instance that conform to CICS requirements.
- “Improving DB2 performance” on page 38 identifies some DB2 attributes and configurations that can be set to improve DB2 performance.

- “Security considerations for DB2 file management” on page 40 discusses how you can restrict access to your DB2 database.

Chapter 5, “Working with CICS queues,” on page 59 discusses how to add queues for use of the DB2 file manager. “Creating the resource definitions for adding queues to the DB2 file manager” on page 62 discusses resource definitions for queues in DB2. “Configuring resources and DB2 with cicsdb2conf” on page 63 describes how to set the required structures for the resource definitions, and how use the CICS resource management command **cicsdb2conf** to create the tables in DB2 that are used for queue management. (This is done automatically if you use **cicscp** to configure the region.)

Chapter 10, “Working with application data files in CICS,” on page 133 discusses adding the data files to the DB2 file manager in “Configuring the File Definitions (FD)” on page 133.

How a CICS system uses a DB2 file manager

The *TXSeries for Multiplatforms Installation Guide* describes how to create a DB2 database that acts as a file manager for a CICS system. If you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, you have created a default file manager to manage the region’s queues and the user’s data files.

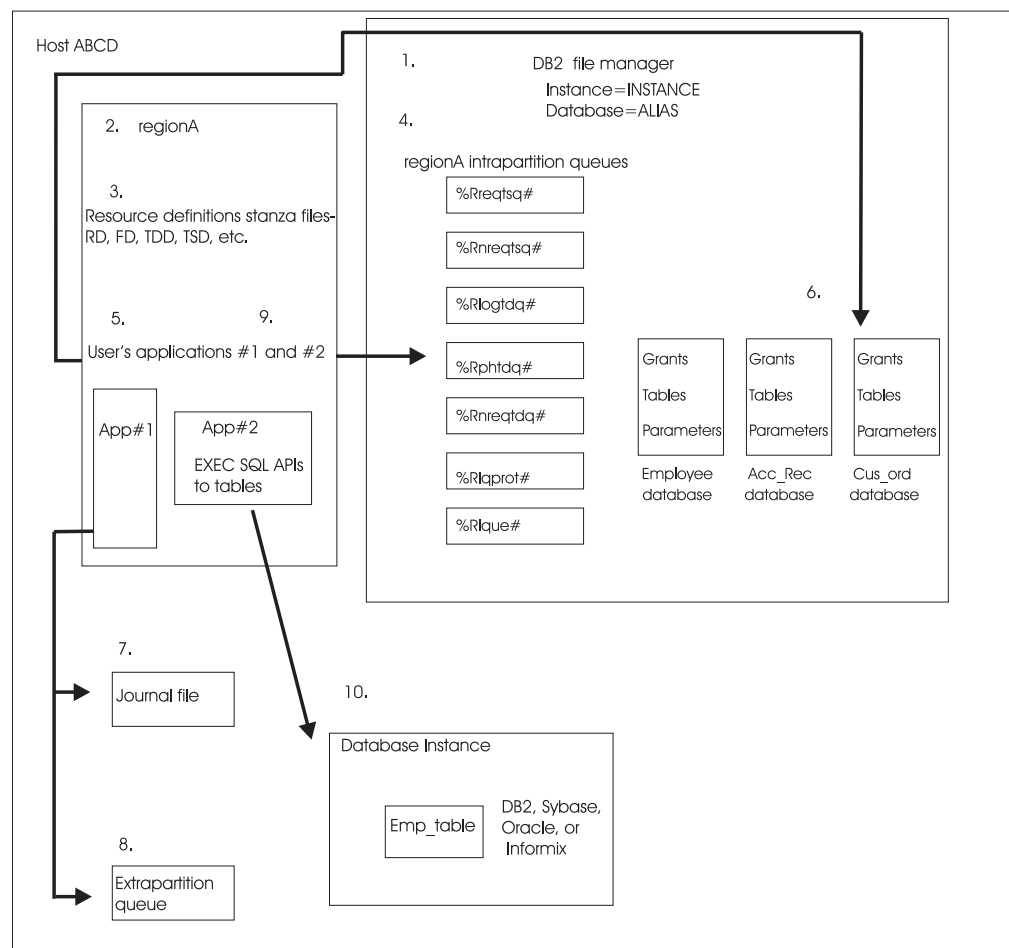


Figure 4. An example of a DB2 file management system for CICS

Figure 4 on page 32 shows an example of CICS file management system that uses a DB2 database as the file manager. In this example, the base configuration, (a file manager, a region, and the intrapartition queues), can be built by using a combination of DB2 commands and **cicscp** commands or CICS resource management commands. Access to the user's application data files, the extrapartition queue, the journal file, and the database tables is enabled through resource definitions that are added to the region's resource definition stanza files.

Figure 4 on page 32 shows how a CICS system uses files and queues. The example CICS system that is shown resides on a host machine that is named hostABCD. A DB2 database, named ALIAS, is acting as the file manager for this system. This DB2 file manager and instance, named INSTANCE, is managing files and queues for regionA. In this example, the user has created an application, App#1, on regionA that stores its data in DB2 tables that are managed by the DB2 file manager, INSTANCE. This application also makes use of a journal file and an extrapartition queue, neither of which is managed by the DB2 file manager. Instead, these files are managed by the operating system file structure.

In addition, on regionA, the user is using a relational database application, App#2, to access data that is stored in tables. The relational database product can be any of the four that are supported by CICS: DB2, Oracle, Sybase, or Informix. The DB2 file manager does not manage the relational database's tables. That job is done by the database product itself.

This section uses the numbered areas that are shown in Figure 4 on page 32, to discuss the ways in which a CICS system, which is created by CICS control program commands, can use files and queues. As the discussion progresses through each numbered area, the pertinent CICS commands, resource definitions, and DB2 commands that are related to the area are described.

1. The DB2 file manager, which is an instance that is identified as INSTANCE, is shown in Area 1. The file manager must exist and must be running before the region that is to use it is created. The file manager can be created by issuing the DB2 commands that are described in "Preparing for DB2 file and queue management" on page 37.
2. When the file manager is running, the region that is to use it can be created. Area 2 in Figure 4 on page 32 shows a region named regionA. If the user follows the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, the command **cicscp -v create region regionA -o INSTANCE -a ALIAS** is issued. The **-o** option is required to identify the instance; the **-a** option is required to identify the database; the **-v** option provides for verbose output from the command. When the user issues this command, the resource definition stanza files, shown in Area 3, and the intrapartition queues, shown in Area 4, are automatically created. The **cicscp start region regionA** command activates the region and connects it to the DB2 file manager.
3. Area 3 in Figure 2 on page 24 shows the file structure that is created to hold the resource definitions for regionA. The resource definition file structure is automatically created when the **cicscp create region** command is issued. The user adds a resource definition entry for each resource that is available to the CICS system. (Chapter 8, "Working with CICS resource definitions," on page 115 identifies the classes of resource definitions and describes the CICS resource management commands that are used to create these definitions.)
4. Area 4 shows the intrapartition queues that are automatically created by the **cicscp create region** command. The **%R** prefix that is attached to each queue name expands to the name of the region. In Figure 4 on page 32, it expands to

regionA. The number sign (#) suffix that is attached to each queue name indicates a table in the DB2 database. Of these seven queues, only the **%Rnreqtdq#** queue can be accessed without the addition of a resource definition entry to identify it to the CICS system. The **%Rlogtdq#**, the **%Rphtdq#**, the **%Rlque**, and the **%Rlqprot#** queues must be named in a Transient Data Definition (TDD) stanza file. The **%Rnreqtsq** and the **%Rreqtsq** queues must be declared in a Temporary Storage Definition (TSD) stanza file. When the TDD or TSD entry is made, CICS can use the CICS application programming interface (API) commands to access these queues.

5. Area 5 shows an application, App#1, that uses EXEC CICS APIs to access DB2 tables that are managed by the DB2 file manager. The application is identified to CICS through a Program Definition (PD) entry in the PD stanza file.
6. Area 6 shows the DB2 tables that hold the data for the user's application, App#1. The application's data tables are defined to CICS through a File Definition (FD), and accessed through EXEC CICS API commands in the application.
7. Area 7 shows a journal file that is used by App#1, which is identified in Area 5. A journal file is not managed by the file manager. It is an ASCII file that is managed by the operating system's file structure. It is identified to CICS by a Journal Definitions (JD) entry, and accessed through EXEC CICS API commands that are embedded in the application.
8. Area 8 shows an extrapartition queue that is used by App#1, which is identified in Area five. An extrapartition queue is not managed by the file manager. It is an ASCII file that is managed by the operating system's file structure. It is identified to CICS by a TDD entry and accessed through EXEC CICS API commands that are embedded in the application.
9. Area 9 shows a relational database application, which the user creates, that resides on regionA. This application uses EXEC SQL commands to access database tables that reside in the operating system's file structure. The application is identified to CICS through a Program Definition (PD) entry in regionA's PD stanza file.
10. Area 10 show the database tables that hold the data for the application that is identified in area nine. These tables are managed entirely by the database product. FD entries are needed to identify these tables to CICS.

Configuring a DB2 database

Configure DB2 as the file manager for a CICS region by using the instructions that are given in this section. Install the required DB2 software before proceeding.

To create a DB2 database:

1. Ensure that you are logged in as **root**.
2. Make the users **root** and **cics** members of the DB2 administrators group.
3. Create and start a DB2 instance to store CICS file management data. See your DB2 documentation for instructions.
4. Set the DB2 database manager TP_MON_NAME configuration parameter to CICS. Enter the command:

```
# db2
db2 => UPDATE DATABASE MANAGER CONFIGURATION USING TP_MON_NAME CICS
```

By setting this parameter, you ensure that DB2 uses the correct binary file for making XA dynamic registration calls.

5. Set the value of the DB2INSTANCE environment variable to the name of DB2 instance that contains the database for CICS file management. Enter the command:

```
# export DB2INSTANCE=instance_name
```

where *instance_name* is the name of the DB2 instance that contains the database for CICS file management.

This command assumes that you are using the Korn shell; if you are using a different shell, change the command as necessary.

6. Ensure that the environment variable LD_LIBRARY_PATH includes the location of the DB2 libraries. Enter the command:

```
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:DB2_install_dir/v8.1/lib
```

where *DB2_install_dir* is the installation directory for DB2.

7. Source the instance owner's db2profile or db2cshrc file to ensure that the appropriate environment variables are set. For example, in the Korn shell, enter the command:

```
# . /home/instance_owner/sql1lib/db2profile
```

where *instance_owner* is the name of the owner of the DB2 instance (which is the same as the instance name).

It is recommended that the environment variables that are in this script are added to the **root** user's .profile, .cshrc, or .login file. On AIX you can also set these environment variables in the /etc/environment file to make them accessible to all users in the Korn shell.

8. Create the DB2 database by using one of these two methods:

- Enter this command if you are using Database Managed Space (DMS) table space:

```
# db2
db2=> CREATE DATABASE database_name user tablespace managed by database \
using (device '/dev/rdsk/partition' #pages)
```

- Enter this command if you are using System Managed Space (SMS) table space:

```
# db2
db2=> CREATE DATABASE database_name
```

where:

- *database_name* is the name of the database that the region is to use
- *partition* is the name of the raw device for the table space volume
- #*pages* is the number of pages to allocate for the table space

9. Connect to the database. Enter the command:

```
db2 => CONNECT TO database_name
```

where *database_name* is the name of the database.

10. Grant system administration authority for the database to the user cics. Enter the command:

```
db2 => GRANT CONNECT ON DATABASE TO USER CICS
```

Use the **DB2 GRANT** command to set the appropriate privileges for other CICS users who must access the database.

11. Exit from DB2, shut down the DB2 instance, and restart the DB2 instance. Enter the commands:

```
db2 => CONNECT RESET
```

```
db2 => TERMINATE
# db2stop
# db2start
```

12. Log into the machine as **root** and perform the following steps if the CICS region that is using the DB2 database is on a different machine:
 - a. Ensure that the DB2 software is correctly installed on the region machine.
 - b. Define the remote machine that contains the DB2 database. Enter the commands:

```
# db2
db2=> CATALOG TCPIP NODE node_name REMOTE IP_address SERVER port
```

where:

- *node_name* is the name of the remote machine that contains the database
 - *IP_address* is the IP address for the remote machine that contains the database
 - *port* is the name of the connection port for the DB2 instance that is on the remote machine that contains the database (as listed in the /etc/services directory of the remote machine)
- c. Enable the region machine to access the remote DB2 database. Enter the command:

```
db2=> CATALOG DATABASE database_name AT NODE node_name \
AUTHENTICATION auth_type
```

where:

- *database_name* is the name of the DB2 database on the remote machine
 - *node_name* is the name of the node that you used in step 12b
 - *auth_type* is the type of authentication that the DB2 database uses
- d. If the DB2 database uses server authentication, set the CICSDB2CONF_CONNECT_USER environment variable to the name of the DB2 instance owner and the CICSDB2CONF_CONNECT_USING environment variable to the password for the DB2 instance owner.

13. Exit from DB2. Enter the commands:

```
db2 => CONNECT RESET
db2 => TERMINATE
```

14. Go to “Configuring and starting a CICS region that uses DB2.”

Configuring and starting a CICS region that uses DB2

A CICS region uses a file manager to store and manage the queues that each region requires. This section describes how to create these queues as database tables in a DB2 database. The queues are:

- The recoverable auxiliary temporary storage queues
- The nonrecoverable auxiliary temporary storage queues
- The logically recoverable transient data queues
- The physically recoverable transient data queues
- The nonrecoverable transient data queues
- The locally queued protected start requests
- The locally queued unprotected start requests

The following procedure configures a CICS region by creating the required queues and files for a DB2 database that is to be used for file management.

Perform these steps on the machine that contains a region that uses DB2:

1. Ensure that you are logged in as **root**.
2. Verify that the DB2 database is running, and that the DB2INSTANCE environment variable is set as described in step 5 on page 35.
3. Configure the region. Enter the **cicscp create region** command:

```
# cics_install_dir/bin/cicscp -v create region region_name -o \
instance_name -a database_name
```

where:

- *region_name* is the name of the region you want to create
- *instance_name* is the name of the DB2 instance
- *database_name* is the name of the DB2 database

Note: The name of your region must not exceed eight characters in length. Do not use underscores (`_`), spaces, or the word "stanza" in the region name. Also, if you intend to connect the region to a network, ensure that the region name meets these requirements:

- It is in uppercase letters if it is to be attached to an SNA network.
- It is unique among all CICS systems connected to the network via TCP/IP or SNA.
- It follows the naming conventions of the network.

4. Start the region. Enter the **cicscp start region** command:

```
# cics_install_dir/bin/cicscp -v start region region_name
```

where *region_name* is the name of the region that you want to start.

Preparing for DB2 file and queue management

"Setting up DB2 for file and queue management" on page 38 describes how to set up a DB2 database so that it can be configured for CICS file management. Before doing the given procedure, do the following:

- For release-specific information that possibly includes changes to the following procedure, see the *TXSeries for Multiplatforms Release Notes*.
- Ensure that you have the following DB2 documentation:
 - The *DB2: Command Reference*
 - The *DB2: Administration Guide*
 - The *DB2: Installation Guide*
- Ensure that DB2 is installed at the levels that CICS supports.
- Ensure that the DB2 instance is set up and started. See the *DB2: Administration Guide*.
- Ensure that the database instance owner is in the group **cics**.

If DB2 returns an XA resource manager error at runtime (which possibly indicates a configuration error), CICS adds the error to the CICS **console.nnnnnnn** file.

Where only one DB2 database is defined to the region, CICS establishes implicit connections to it. All file control, queue, and SQL operations are directed to that database. SQL-based transactions that access DB2 databases other than those that are supported in the file system must be coded to ensure that the connection to the file system database is maintained across CICS file control, queue, and transaction control commands. For example, it might be necessary to code an explicit EXEC SQL CONNECT statement before you issue some CICS file control, queue, and transaction control commands.

Setting up DB2 for file and queue management

Follow this procedure to set up a DB2 database so that it can be configured for CICS file management.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. **On the AIX platform.** Use the following commands to create and link the DB2 shared object that is referenced by the CICS COBOL runtime, CICS transactions, and the DB2 switch load file at runtime:

```
cd /usr/lpp/cics/DB2Version/lib
ar -vx libdb2.a
mv shr.o db2.o
ln -s /usr/lpp/cics/DB2Version/lib/db2.o /usr/lib/db2.o
```

where *DB2Version* specifies the release of DB2 that you are using.

3. Create a DB2 database for CICS by using the **db2 CREATE DATABASE database** command. (This step can be omitted if you want to use an existing database. This step assumes you are using SMS Table Space. If you are using DMS Table Space, see “Improving DB2 performance” for an appropriate replacement for this step.)
`db2 create database database`
4. Enter the command **db2** to access the DB2 command-line processor. Connect to the database by using the **CONNECT TO database** command.
5. Grant the appropriate privileges on the new database. For example, if you plan to allow CICS to connect implicitly to the database as user **cics**, and if user **cics** does not have SYSADM or DBADM authority in the database, use the **GRANT CONNECT ON DATABASE TO USER CICS** command to grant the appropriate privilege. Enter **quit** to leave the DB2 command-line processor.

Note: DB2 tables and their associated indexes that map to CICS files need to be owned by the DB2 user **cics**.

6. Run the **cicsdefaultservers** command to set up the file system that contains the default schemas that define data files to DB2. Refer to “Setting up the data files on DB2 with the cicsdb2import command” on page 138 to find out how the schematics are used.

See the *DB2: Command Reference* for information about the Command Line Processor (CLP) **db2** command, the Start DB2 (**db2start**) command, the **create database** command, and the **grant** command.

Improving DB2 performance

DB2 provides a wide range of configuration parameters that can improve DB2’s performance in the complex environments where DB2 databases need to interact. This section offers only a few of the many possible customizations that you can use in your environment. Refer to DB2 documentation for more information about parameters that can be set to improve the performance of DB2 in your environment.

Optimizing the use of DB2 table space

DB2 provides a choice of two types of table space:

- System Managed Space (SMS) Table Space, in which operating system file manager calls are used to control the storage space.
- Database Managed Space (DMS) Table Space, in which the database manager controls the storage space.

By default, DB2 uses SMS Table Space. It is recommended, however, that DMS Table Space is used with CICS because it generally provides a performance improvement.

If you are using DMS Table Space, do the following:

1. Create the location of the physical table space:
 - **On the AIX or HP-UX platform.** Create the logical volume that is to be used for the table space.
 - **On the Solaris platform.** Create the partition that is to be used for the table space.
2. Use the **tablespace** option when you create the database.

For example, the following commands create a logical volume that has 10 logical partitions and a table space with 10,000 pages:

On the AIX platform.

```
mklv -y logvolABC rootvg 10
chown cics:cics /dev/logvolABC /dev/rlogvolABC
db2start
db2 create database CHARLEY user tablespace \
    managed by database using (device '/dev/rlogvolABC' 10000)
```

See the *DB2: Administration Guide* for details about how to create DMS Table Space.

Improving lock performance in DB2

Using DB2 file control increases the risk of deadlock because all queues are handled as recoverable. To help avoid deadlocks, define the Transaction Definition attribute **DeadLockTimeout**. The default is 0, which means that no timeout exists.

If you want to specify an operational lock timeout value or reconfigure the deadlock timeout interval for DB2 queue and file management, set the LOCKTIMEOUT and DLCHKTIME parameters for the database. By default, no operational lock timeout value is set.

The appropriate place in the configuration process to set these parameters is after you have exited the DB2 command line processor following the create, connect, and grant database procedures. For more information about configuration parameters, see the *DB2: Administration Guide*.

Avoiding deadlocks by turning off next key locking: Try using the DB2 variable DB2_RR_TO_RS to switch off next key locking. This can avoid unwanted deadlocks, but if this variable is used, DB2 will acquire more internal locks. In this case, the locklist or maxlocks database parameters possibly need to be increased to accommodate the extra internal locks. If the locklist parameter is not set high enough, lock escalation occurs and concurrency suffers.

Set the DB2_RR_TO_RS environment variable to YES before using **db2start** to start the database manager. This changes the behavior of the DB2 index manager to avoid next key locks, which can provide some performance benefits to a CICS region that is using DB2 for queue and file management.

However, setting DB2_RR_TO_RS to YES requires that you avoid the use of repeatable-read cursors in any DB2 program that uses this DB2 instance. A region that uses DB2 for queue and file management does not use this type of cursor. Because of this restriction, it is advisable to set this environment variable only for instances that are exclusively for CICS use.

Dynamic registration of the transaction processor

Also, it is recommended that you set the DB2 database manager TP_MON_NAME database configuration parameter to **CICS**. This parameter setting dynamically identifies CICS as the transaction processor.

Security considerations for DB2 file management

In CICS, you can define resources so that only specific user IDs have access to them. This is described in “Authorizing user access to CICS resources” on page 99.

You can further restrict access to the DB2 database by using the **cicsdb2conf -g** option as described in the *TXSeries for Multiplatforms Administration Reference*.

To use DB2 for queue and file management, the user ID under which CICS connects to DB2 must have at least CONNECT privilege to the appropriate database.

See the *DB2: Administration Guide* for more information.

Note: *Privileges* enable users to create or access database resources. *Authority* levels provide a method of grouping privileges and higher-level operations for maintaining database managers and utility operations. Together, these act to control access to the database manager and its database objects. Users can access only those objects for which they have the appropriate *authorization*; that is, the required privilege or authority.

DB2 allows connections from CICS that are implicitly authenticated as user **cics**, or explicitly authenticated as some other user. The choice of implicit or explicit authentication depends on your DB2 environment. For example, if CICS and the DB2 server are installed on the same machine, implicit authentication through user **cics** is readily supported.

An authenticated DB2 user can have different levels of authorization in a DB2 database. For example, under DB2, the user **cics** can be configured to have system administration (SYSADM) or database administration (DBADM) authority, in addition to any number of privileges, authorities, and authorizations.

On the AIX platform. If multiple users want to have access to external file handler applications on the AIX platform, they must be given execute access to the following files in the **/usr/lpp/cics/etc** directory:

- **cics101e**
- **cics101f**
- **cics101j**
- **cics101l**
- **cics101s**
- **cics101u**

Use the following command to grant this access:

```
db2 grant execute on package buildadm.package name to user ID
```

Administration of file and queue management on Oracle

This section contains the following topics:

- “How a CICS system uses an Oracle file manager” discusses the creation of an example region that uses an Oracle database as its file manager.
- “Configuring an Oracle database” on page 44 describes how to configure Oracle as the file manager for a CICS region.
- “Preparing for Oracle file and queue management” on page 47 describes the creation of the database and instance that conform to CICS requirements.
- “Improving Oracle performance” on page 49 identifies some of the Oracle attributes and configurations that can be set to improve Oracle performance.
- “Security considerations for Oracle file management” on page 49 discusses how you can restrict access to your Oracle database.

You can find more information about the Oracle file manager in the following chapters:

- Chapter 5, “Working with CICS queues,” on page 59 discusses how to add queues for use of the Oracle file manager. “Creating the resource definitions for adding queues to the Oracle file manager” on page 64 discusses resource definitions for queues in Oracle. “Configuring resources and Oracle with `cicsoraconf`” on page 64 describes how to set the required structures for the resource definitions and how to create the tables in Oracle that are used for queue management by using the CICS resource management command `cicsoraconf`. (This is done automatically if you use `cicscp` to configure the region.)
- Chapter 10, “Working with application data files in CICS,” on page 133 discusses adding the data files to the Oracle file manager in “Configuring the File Definitions (FD)” on page 133.

How a CICS system uses an Oracle file manager

The *TXSeries for Multiplatforms Installation Guide* describes how to create an Oracle database that acts as a file manager for a CICS system. If you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, you have created a default file manager to manage the region’s queues and the user’s data files.

Figure 5 on page 42 shows an example of CICS file management system that uses an Oracle database as the file manager. In this example, the base configuration, (a file manager, a region, and the intrapartition queues), can be built by using a combination of Oracle commands and `cicscp` commands or CICS resource management commands. Access to the user’s application data files, the extrapartition queue, the journal file, and the database tables is enabled through resource definitions that are added to the region’s resource definition stanza files.

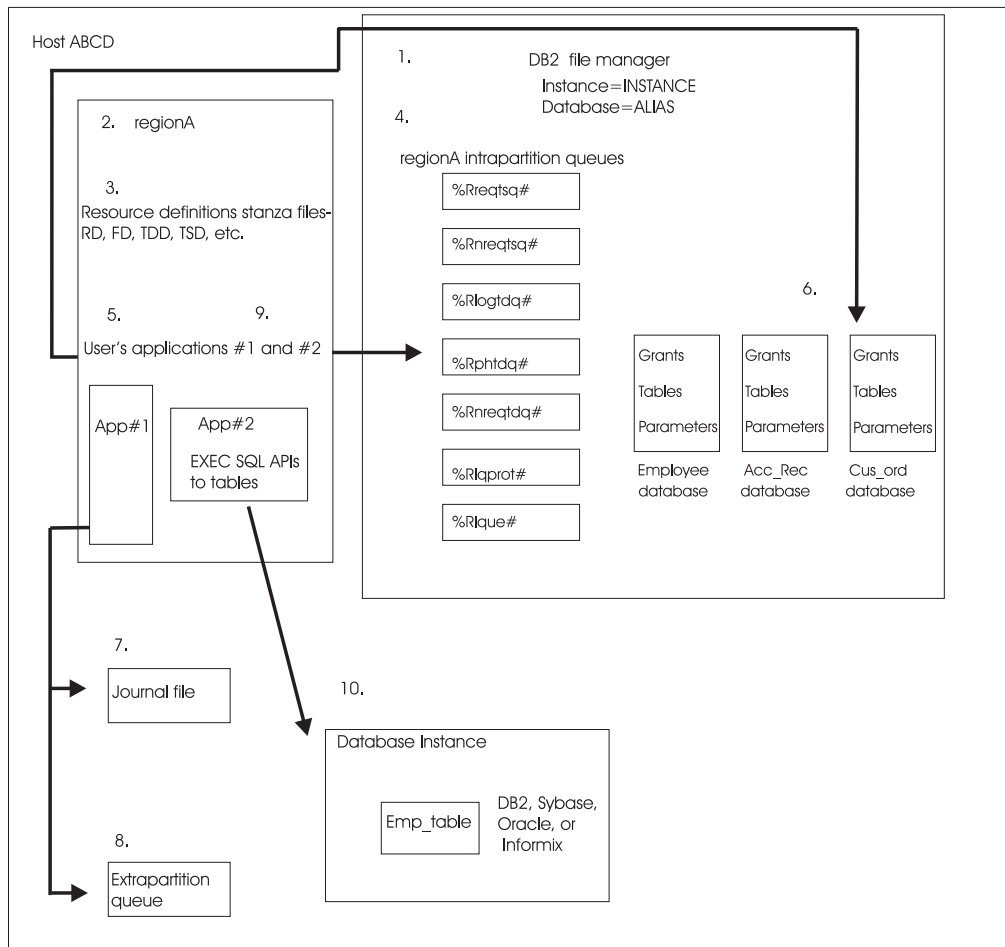


Figure 5. An example of an Oracle file management system for CICS

Figure 5 shows how a CICS system uses files and queues. The example CICS system that is shown in this diagram resides on a host machine that is identified as hostABCD. An Oracle database, named SAMPLE, is acting as the file manager for this system. This Oracle file manager and instance, named ORASID, is managing files and queues for regionA. In this example, the user has created an application, App#1, on regionA that stores its data in Oracle tables that are managed by the Oracle file manager, ORASID. This application also uses a journal file and an extrapartition queue, neither of which is managed by the Oracle file manager. Instead, these files are managed by the operating system file structure.

In addition, on regionA, the user is using a relational database application, App#2, to access data that is stored in tables. The relational database product can be any of the four that are supported by CICS: DB2, Oracle, Sybase, or Informix. The Oracle file manager does not manage the relational database's tables. That job is done by the database product itself.

This section uses the numbered areas that are shown in Figure 5, to discuss the ways in which a CICS system, created by CICS control program commands, can make use of files and queues. As the discussion progresses through each numbered area, the pertinent CICS commands, resource definitions, and Oracle commands that are related to the area are described.

1. The Oracle file manager, an instance identified as ORASID, is shown in Area 1. The file manager must exist and must be running before the region that is

to use it is created. The file manager can be created by issuing the Oracle commands that are described in “Preparing for Oracle file and queue management” on page 47.

2. When the file manager is running, the region that is to use it can be created. Area 2 in Figure 5 on page 42 shows a region named regionA. If the user follows the instructions that are given in the *TXSeries for Multiplatforms Installation Guide*, the command **cicscp -v create region regionA -o ORACLE -u cics/cics** is issued. The **-o** option is required to identify the instance; the default database for the instance ORASID is used; the **-v** option provides for verbose output from the command. When the user issues this command, the resource definition stanza files, shown in Area 3, and the intrapartition queues, shown in Area 4, are automatically created. The **cicscp start region regionA** command activates the region and connects it to the Oracle file manager.
3. Area 3 in Figure 5 on page 42 shows the file structure that is created to hold the resource definitions for regionA. The resource definition file structure is automatically created when the **cicscp create region** command is issued. The user adds a resource definition entry for each resource that is available to the CICS system. (Chapter 8, “Working with CICS resource definitions,” on page 115 identifies the classes of resource definitions and describes the CICS resource management commands that are used to create these definitions.)
4. Area 4 shows the intrapartition queues that are automatically created by the **cicscp create region** command. The **%R** prefix that is attached to each queue name expands to the name of the region. In Figure 5 on page 42, it expands to regionA. The number sign (#) suffix that is attached to each queue name indicates a table in the Oracle database. Of these seven queues, only the **%Rnreqtdq#** queue can be accessed without the addition of a resource definition entry to identify it to the CICS system. The **%Rlogtdq#**, the **%Rphtdq#**, the **%Rlque**, and the **%Rlqprot#** queues must be named in a Transient Data Definition (TDD) stanza file. The **%Rnreqtsq** and the **%Rreqtsq** queues must be declared in a Temporary Storage Definition (TSD) stanza file. When the TDD or TSD entry is made, CICS can use the CICS application programming interface (API) commands to access these queues.
5. Area 5 shows an application, App#1, that uses EXEC CICS APIs to access Oracle tables that are managed by the Oracle file manager. The application is identified to CICS through a Program Definition (PD) entry in the PD stanza file.
6. Area 6 shows the Oracle tables that hold the data for the user’s application, App#1. The application’s data tables are defined to CICS through a File Definition (FD) and accessed through EXEC CICS API commands in the application.
7. Area 7 shows a journal file that is used by App#1, identified in Area 5. A journal file is not managed by the file manager. It is an ASCII file that is managed by the operating system’s file structure. It is identified to CICS by a Journal Definitions (JD) entry and accessed through EXEC CICS API commands that are embedded in the application.
8. Area 8 shows an extrapartition queue that is used by App#1, which is identified in Area five. An extrapartition queue is not managed by the file manager. It is an ASCII file that is managed by the operating system’s file structure. It is identified to CICS by a TDD entry and accessed through EXEC CICS API commands that are embedded in the application.
9. Area 9 shows a relational database application, created by the user, that resides on regionA. This application uses EXEC SQL commands to access

database tables that reside in the operating system's file structure. The application is identified to CICS through a Program Definition (PD) entry in regionA's PD stanza file.

10. Area 10 show the database tables that hold the data for the application identified in area nine. These tables are managed entirely by the database product. FD entries are needed to identify these tables to CICS.

Configuring an Oracle database

If you intend to use a local Oracle database as the file manager for a CICS region, you must configure Oracle on the machine on which you want it to run. Before using these instructions, install the required Oracle software.

You can also configure a CICS region to use a Oracle database that is on a remote machine. See "Configuring a remote Oracle database" on page 45 for this information.

To configure a local Oracle database for use by a CICS region, perform the following steps on the machine on which you want the database to run:

1. Ensure that you are logged in as **root**.
2. Make the users **root** and **cics** members of the Oracle administrators group. In addition, if the user **root** has connect and dba privileges, all commands can be issued while you are logged in as **root**. See your Oracle documentation for information about this group.
3. If you have not done so already, create and start the Oracle instance that is to contain the database for CICS file management. To issue these commands, you must be logged in as a Oracle user who is part of an administrative group. See your Oracle documentation for instructions.
4. Set the value of the ORACLE_SID environment variable to the name of the Oracle instance that contains the database for CICS file management:

```
#export ORACLE_BASE=/oracle/9202
#export ORACLE_HOME=/oracle/9202/ORASID
#export ORACLE_SID=ORASID
#export ORA_CLIENT_LIB=shared
#export PATH=$PATH:$ORACLE_HOME/bin
#export LIBPATH=$LIBPATH:$ORACLE_HOME/lib32
```

Set these variables both for the user **oracle** and for the user **root**.

Note: This example assumes that you are using the Korn shell; if you are using a different shell, change the command as necessary. This example also assumes that:

- You set the Oracle instance to ORASID before you installed the Oracle software
- You have used the configuration assistants to set up a database
- The database belongs to ORASID

It is recommended that you add the command to execute this script to the **root** user's .profile file that is in the Korn shell, or to the .login file or .cshrc file that is in the C shell. In the Korn shell, you can also set the environment variables in this script in the /etc/environment file to make them accessible to all users.

5. Login as user **oracle**.
6. Connect to the database. Enter the command:

```
sqlplus => (login as system/manager)
sqlplus => create user cics identified by cics default tablespace users quota
unlimited on users
```


7. Grant the user CICS permission to connect to the database. Enter the command:

```
sqlplus => GRANT CONNECT,RESOURCE TO CICS
sqlplus => quit
```
8. Use the Oracle GRANT command to set the appropriate privileges for other CICS users who must access the database.

Proceed to “Configuring and starting a CICS region that uses an Oracle database.” For more information about how to configure Oracle, see your Oracle documentation.

Configuring a remote Oracle database

CICS regions can use Oracle databases that are on other machines. If the CICS region is on an Oracle client machine and uses a database that is located on an Oracle server machine, log into the client machine as a user who is a member of the Oracle administration group create the local net service for the remote database. You can use the Oracle utility netca to create the local net service name:

1. Invoke the oracle utility netca.
2. Select **Local net service name configuration**.
3. Select the option **Add**.
4. Select the oracle8i or database or service.
5. Enter the remote database service name (remote database SID). For Example Ora817 or Oracl92.
6. Select the TCP protocol.
7. Enter the remote database host name or IP address
8. Select **Yes** to perform the connectivity test. It should show Connecting...Test Successful.
9. Save this configuration to a local Net Service name. For Example ora92.
10. Create the region on the remote machine. Enter the command:

```
Cicscp -v create region RgnName -o ORACLE -u cics@ora92/cics
```

Go to “Configuring and starting a CICS region that uses an Oracle database.” For more information about how to configure Oracle, see your Oracle documentation.

Configuring and starting a CICS region that uses an Oracle database

A CICS region uses a file manager to store and manage the queues that each region requires. This section describes how to create these queues as database tables in a local Oracle database. (If you want to use a remote Oracle database, follow the instructions given in “Configuring and starting a CICS region that uses a remote Oracle database” on page 46.) The queues are:

- The recoverable auxiliary temporary storage queues
- The nonrecoverable auxiliary temporary storage queues
- The logically recoverable transient data queues
- The physically recoverable transient data queues
- The nonrecoverable transient data queues
- The locally queued protected start requests
- The locally queued unprotected start requests

The following procedure configures a CICS region by creating the required queues and files for an Oracle database that is to be used for file management.

If you plan to use a local Oracle database as your region's file manager, perform these steps on the machine that contains the region:

1. Ensure that you are logged in as **root**.
2. Verify that the Oracle database is running, and that the ORACLE_HOME and ORACLE_SID environment variable is set.
3. Configure the region. Enter the **cicscp create region** command:

```
# cics_install_dir/bin/cicscp -v create region region_name -o \
ORACLE -u user/password
```

where:

- *region_name* is the name of the region you want to create
- -o ORACLE is mandatory
- *user/password* is for the **cics** user who is registered with the Oracle database

Note: The name of your region must not exceed eight characters in length. Do not use underscore characters (_), spaces, or the word "stanza" in the region name. Also, if you intend to connect the region to a network, ensure that the region name meets these requirements:

- It is in uppercase letters if it is to be attached to a Systems Network Architecture (SNA) network.
- It is unique among all CICS systems connected to the network by way of TCP/IP or SNA.
- It follows the naming conventions of the network.

4. Start the region. Enter the **cicscp start region** command:

```
# cics_install_dir/bin/cicscp -v start region region_name
```

where *region_name* is the name of the region that you want to start.

Configuring and starting a CICS region that uses a remote Oracle database:

After you have set up the Oracle client and server machines to enable CICS to access a remote Oracle database (as described in "Configuring a remote Oracle database" on page 45), perform these steps to configure and start a region that uses that database:

1. Ensure that you are logged in as **root**.
2. Configure the region. Enter the **cicscp create region** command:

```
# cics_install_dir/bin/cicscp -v create region region_name -o \
ORACLE -u user@net_service/password
```

where:

- *region_name* is the name of the region that you want to create
- -o ORACLE is mandatory
- *user/password* is for the **cics** user who is registered with the Oracle database
- *net_service* is the local net service configured for the remote database

Note: The name of your region must not exceed eight characters in length. Do not use underscore characters (_), spaces, or the word "stanza" in the region name. Also, if you intend to connect the region to a network, ensure that the region name meets these requirements:

- It is in uppercase letters if it is to be attached to a Systems Network Architecture (SNA) network.

- It is unique among all CICS systems that are connected to the network by way of TCP/IP or SNA.
 - It follows the naming conventions of the network.
3. Update the region XAD definition for the database.
 4. Start the region. Enter the **cicscp start region** command:

```
# cics_install_dir/bin/cicscp -v start region region_name
```

where *region_name* is the name of the region that you want to start.

Preparing for Oracle file and queue management

“Setting up Oracle for file and queue management” describes how to set up an Oracle database so that it can be configured for CICS file management.

Before using this procedure, do the following:

- For release-specific information that possibly includes changes to the following procedure, see the *TXSeries for Multiplatforms Release Notes*
- Ensure that you have the Oracle documentation.
- Ensure that Oracle is installed at the levels supported by CICS.
- Ensure that the Oracle instance is set up and started. See the Oracle documentation, or use the configuration assistants to set up an instance with a default database.
- Ensure that the ID “oracle” is in the group **cics**.

If Oracle returns an XA resource manager error at runtime (which possibly indicates a configuration error), CICS adds the error to the CICS **console.nnnnnnn** file.

Where only one Oracle database is defined to the region, CICS establishes implicit connections to it. All file control, queue, and SQL operations are directed to that database. SQL-based transactions that access Oracle databases other than those that are supported in the file system must be coded to ensure that the connection to the file system database is maintained across CICS file control, queue, and transaction control commands. For example, it might be necessary to code an explicit EXEC SQL CONNECT statement before issuing some CICS file control, queue, and transaction control commands.

Setting up Oracle for file and queue management

Follow this procedure to set up an Oracle database so that it can be configured for CICS file management.

Note: This procedure assumes that you are using Oracle version 9.2.0.2 or higher.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Create a new filesystem mounted on /DBMS/oracle.
3. Make a directory called /DBMS/oracle/9202.
4. Create a user ID named **oracle** (with home directory as /DBMS/oracle/9202).
5. Create a group named **dba**. Ensure that user **oracle** is part of group **dba** and **cics**. Also ensure that users **cics** and **root** are part of the group **dba**.
6. Login as user **oracle**.

The profile for the user **oracle** should have the following environment (assuming default shell as ksh):

```
export ORACLE_BASE=/oracle/9202
export ORACLE_SID=ORASID
export ORACLE_HOME=$ORACLE_BASE/$ORACLE_SID
export PATH=$PATH:$ORACLE_HOME/bin
export LIBPATH=$LIBPATH:$ORACLE_HOME/lib32 (if a 64-bit Oracle installation)
export ORA_CLIENT_LIB=shared
```

(The LIBPATH variable can be changed to LD_LIBRARY_PATH on Solaris, and SHLIB_PATH on HP-UX)

7. Install Oracle, version 9.2.0.2 server and client. Configuration assistants that are packaged with Oracle can help you to create a database. Here, it is assumed that the default database is named "sample" and that the ORACLE_SID is set to "ORASID". The oracle processes that belong to the instance identifier ORASID are started automatically.

Here is an example of the Oracle processes that could be running on your system, depending upon the components that are installed:

```
$ ps -ef | grep ORASID
oracle 23258 1 0 Jun 06 - 0:16 ora_lgwr_ORASID
oracle 23822 1 0 Jun 06 - 0:14 ora_dbw0_ORASID
oracle 24266 1 0 Jun 06 - 2:12 ora_ckpt_ORASID
oracle 24712 1 0 Jun 06 - 0:00 /oracle/9202/ORASID/bin/tnslsnr LISTENER -inherit
oracle 24990 1 0 Jun 06 - 0:16 ora_pmon_ORASID
oracle 25134 1 0 Jun 06 - 0:00 ora_reco_ORASID
oracle 25818 1 0 Jun 06 - 0:14 ora_smon_ORASID
```

8. Enter the command **sqlplus** to access the Oracle command-line processor:

```
$ sqlplus
```

```
SQL*Plus: Release 9.2.0.2.0 - Production on Tue Jul 29 11:25:41 2003
```

```
(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Enter user-name: system
```

```
Enter password: manager
```

```
Connected to:
```

```
Oracle8i Enterprise Edition Release 9.2.0.2.0 - Production
```

```
JServer Release 9.2.0.2.0 - Production
```

```
sqlplus => create user cics identified by cics default tablespace users quota
unlimited on user;
```

```
sqlplus => grant connect,resource to cics;
```

```
SQL> quit;
```

Now the Oracle database is ready for file management for a cics region. The first sql statement created a user **cics** with password **cics**. This user can connect to the Oracle database. The next step granted dba privileges to user **cics** on the new database. The level of privileges could be tuned by the administrator. For example, if you plan to allow implicitly CICS to connect to the database as user **cics**, and if user **cics** does not have SYSADM or DBADM authority in the database, use the **GRANT CONNECT TO USER CICS** command to grant the appropriate privilege.

9. Enter quit to leave the Oracle command-line processor.

Note: Oracle tables and their associated indexes that map to CICS files must be owned by the Oracle user **cics**.

10. Run the **cicsdefaultservers** command to set up the file system that contains the default schemas that are used to define data files to Oracle. See "cicsoraimport command - Build metadata for Oracle " in the *TXSeries for Multiplatforms Administration Reference* to find out how the schematics are used.

See the Oracle *Command reference* for information about the Command Line Processor (CLP) **sql** command, and the **grant** command.

Improving Oracle performance

Refer to Oracle documentation for more information about parameters that you can use to tune the performance of Oracle in your environment.

Security considerations for Oracle file management

In CICS, you can define resources so that only specific user IDs have access to them. This is described in “Authorizing user access to CICS resources” on page 99.

You can further restrict access to the Oracle database by using the **cicsoraconf -g** option as described in the *TXSeries for Multiplatforms Administration Reference*.

To use Oracle for queue and file management, the user ID under which CICS connects to Oracle must have at least CONNECT privilege to the appropriate database.

See the *Oracle: Administration Guide* for more information.

Note: *Privileges* enable users to create or access database resources. *Authority* levels provide a method of grouping privileges and higher-level operations for maintaining database managers and utility operations. Together, these act to control access to the database manager and its database objects. Users can access only those objects for which they have the appropriate *authorization*; that is, the required privilege or authority.

Oracle allows connections from CICS that are implicitly authenticated as user **cics**, or explicitly authenticated as some other user. The choice of implicit or explicit authentication depends on your Oracle environment. For example, if CICS and the Oracle server are installed on the same machine, implicit authentication through user **cics** is readily supported.

An authenticated Oracle user can have different levels of authorization in an Oracle database. For example, under Oracle, the user **cics** can be configured to have database administration (DBA) authority, in addition to any number of privileges, authorities, and authorizations.

Migrating file and queue management (DB2 and SFS)

“Moving file management from DB2 to SFS” on page 52 and “Moving file management from SFS to DB2” on page 50 describe how to change CICS queue and file management on an existing region. Review the following information before using these procedures:

- The CICS queue files do not need to be transferred if the region is to be cold started after changing the file manager, or if they are empty. The **cicssdt** and **cicsddt** commands can be used to list the number of records in each file.
- For mass-migration of a large number of files, see the README file that is the directory for your operating system (see Table 10):

Table 10. The location of the README file on the AIX, HP-UX, and Solaris platforms

Platform	Directory
AIX	/usr/lpp/cics/utills/cicsdb2m

Table 10. The location of the README file on the AIX, HP-UX, and Solaris platforms (continued)

Platform	Directory
HP-UX	/opt/cics/utls/cicsdb2m
Solaris	/opt/cics/utls/cicsdb2m

- Although all CICS queues behave as fully recoverable when DB2 is used as the file manager, the recovery attributes, as defined in the Temporary Storage Definitions (TSD) and Transient Data Definitions (TDD) entries for the queues, are not changed. These settings determine which file is used to store data for each queue. Therefore, when migrated from DB2, queues that are defined as nonrecoverable will be nonrecoverable in SFS.
- All records over 4005 bytes are treated as variable by CICS when using DB2 as the file manager.
- Migrated CICS applications that were written to access fixed records that are longer than 4005 bytes will find the following changes in CICS behavior:
 - On a READ, the value that is returned in the length field is the true length of the data, not the expected fixed length.
 - On a READ, the LENGTH option must be specified; otherwise, LENGERR is returned.
- When DB2 is used as the file manager, files that are over 4005 bytes long must be created with the last column of a variable type (VARCHAR, LONGVAR, BLOB). The total length of the preceding fixed columns cannot exceed 4005 bytes.
- The process of migrating CICS files from SFS to DB2 is subject to the following restrictions:
 - SFS files that have a fixed length of over 4005 bytes must be redefined.
 - SFS files that have a fixed length of over 4005 bytes and have key fields beyond byte 4004 cannot be directly migrated. The file must be redesigned to ensure that all key fields are within the first 4005 bytes of the record.
- File locking and concurrency on DB2 differs from file locking and concurrency on SFS. See the *TXSeries for Multiplatforms Application Programming Guide* for more information.

Moving file management from SFS to DB2

The purpose of this procedure is to move CICS queue and file management from SFS to DB2 for a region that was configured to use an SFS server.

Note:

- This procedure creates Product Definitions (XAD) that define the DB2 database to the region. If you already have an XAD defined for this database, either delete it by using the **cicsdelete** command, or use the **-s** option with the **cicsdb2conf** command. Otherwise, two XAD entries can possibly exist for this database and lead to unpredictable results.
- Oracle does not allow particular special characters in the field names. Before migrating SFS files to DB2, refer to the information in the DB2 SQL reference that describes field name restrictions.

This procedure uses the **cicsdb2conf** command to configure the region and the DB2 database for file management. To use DB2 explicit authentication, use the **-u** option to specify the user name and user ID. (Alternatively, you can use the

CICSDB2CONF_CONNECT_USER and CICSDB2CONF_CONNECT_USING environment variables.) If you want to specify the creation of a Product Definitions (XAD) entry for single-phase commit, use the **-n** option. If you do not want to create a Product Definitions (XAD) entry for this database (when one has already been created), use the **-s** option.

To use the **cicsdb2conf** command to configure the region and the DB2 database for file management:

- You must be logged in as **root** on the CICS server host and **root** must be set up as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
 - You must verify that the database has been started.
 - If you are using explicit DB2 authentication to connect CICS to DB2:
 - The user must be set up and the minimum permissions granted. (See the *TXSeries for Multiplatforms Installation Guide* for more information.)
 - If you do not want to be prompted for the password, you must set CICSDB2CONF_CONNECT_USING environment variable to the database user ID password.
1. Shut down the region. Enter the command:


```
cicscp -v stop region regionName
```
 2. Reconfigure the region for DB2. Enter the command:


```
cicsdb2conf -r regionName -C -i instanceName -a databaseName
```
 3. To transfer the files from SFS to DB2:
 - a. Identify the files that the region requires:
 - All files that are defined in the region’s File Definitions (FD). The name of the file is specified with the FD **BaseName** attribute.
 - CICS queues that are defined in the Region Definitions (RD). See Table 11.

Table 11. Queue name values (where %R is the region name)

RD attribute	SFS name (default)	DB2 name (mandatory)
LogicalTDQFile	%Rcicstdqlgfile	%Rlogtdq#
PhysicalTDQFile	%Rcicstdqphfile	%Rphtdq#
NonRecTDQFile	%Rcicstdqnofile	%Rnrectdq#
RecTSQFile	%Rcicsrectsqfil	%Rrectsq#
NonRecTSQFile	%Rcicsnrectsqfile	%Rnrectsq#
LocalQProtectFile	%Rcicsplqfile	%Rlqprot#
LocalQFile	%Rcicsnlqfile	%Rlque#

- b. Transfer the files from SFS to flat file format. Use **cicssdt** with the **stof** option. For example:


```
cicssdt -s serverName -c stof BaseName
```

 Specify **N** in the Dump file Asis? field.


```
[Flat Filename .....: BaseName.sdt
[Dump file Asis ? ..... Y/[N]: N
[Dump Record(s) .....: ALL
```
 - c. Create the user-defined files in DB2.
 - d. Transfer the files from flat file format into DB2. Use the **cicsddt** command with the **ftod** option. For example:


```
cicsddt -s databaseName -c ftod tableName
```

Use the same flat file name that you created in step 3b on page 51. You must type over the whole field. For example:

```
[Flat Filename .....: BaseName.sdt
```

For more information, see the *TXSeries for Multiplatforms Administration Reference*.

Note that the queue files need to be imported under the DB2 names that are shown in Table 11 on page 51. For example, to import the recoverable transient data queue file for regionA, use the following commands (assuming that the SFS server file is using the default name):

- 1) Enter the command:

```
cicssdt -s /./cics/sfs/hostABCD -c stof regionAcicsrectsqfile
```

This command produces a flat file named **regionAcicsrectsqfile.sdt**.

- 2) Enter the command:

```
cicsddt -s cicstest -c ftod regionArectsq#
```

This command prompts you for the name of the flat file.

- 3) Enter:

```
regionAcicsrectsqfile.sdt
```

4. Cold start the region.

Moving file management from DB2 to SFS

This procedure moves CICS queue and file management from DB2 to SFS for a region that was configured to use DB2.

Before using the procedure, do the following:

- Review “Migrating file and queue management (DB2 and SFS)” on page 49 so that you fully understand the effects of this change.
- Note that recovery attributes are ignored when DB2 is used for CICS queue and file management. Verify that the following attributes are set correctly for use with SFS:
 - The Transient Data Definitions (TDD) **RecoveryType** attribute
 - The Temporary Storage Definitions (TSD) **RecoverFlag** attribute
 - The File Definitions (FD) **RecoverStatus** attribute

If the attributes are not set correctly, delete the queues and redefine the attributes; File Definitions (FD) can be changed.

1. Use a normal shutdown to shut down the region. For example:

```
cicscp -v -l /tmp/cicscp.log stop region regionA
```

2. If an SFS server does not exist that this region can use, use the instructions that are given in “Overriding the defaults imposed by the cicscp start sfs_server command” on page 29 to set one up.

3. Transfer the files from DB2 to SFS:

- a. Identify the files that the region requires:

- All files that are defined in the region’s File Definitions (FD). The name of the file is specified with the FD **BaseName** attribute.
- CICS queues that are defined in the Region Definitions (RD). See Table 11 on page 51.

- b. Transfer the files from DB2 to flat file format by using the **cicsddt** command with the **dtof** option. See the *TXSeries for Multiplatforms Administration Reference* for more information.

- c. Create the user-defined files in SFS.
- d. Use the **cicssfscnf** command to configure the SFS server. For example, if the region name is regionA and the SFS server is `./:/cics/sfs/hostABCD`, enter:

```
cicssfscnf -R wc regionA DefaultFileServer=./:/cics/sfs/hostABCD
```

The **-R** option resets the file system to SFS in the Region Definitions (RD) and File Definitions (FD), and renames the queues. Use the **-I** option if the queue files were transferred in a previous step.

Attention: This deletes any system-defined Product Definitions (XAD) for this region.

- e. Transfer the files from flat file format into SFS by using the **cicsddt** command with the **ftos** option. See the *TXSeries for Multiplatforms Administration Reference* for more information.

Note that the queue files must be imported under the SFS server names that are shown in Table 11 on page 51. For example, to import the recoverable TDQ file for regionA, use the following commands (assuming that the SFS server file is using the default name):

- 1) Enter the command:

```
cicsddt -s cicstest -c dtof regionArectsq#
```

This command produces a flat file named **regionArectsq#.ddt**.

- 2) Enter the command:

```
cicssdt -s ./:/cics/sfs/hostABCD -c ftos regionAcicsrectsqfile
```

This command prompts you for the name of the flat file.

- 3) Enter:

```
regionArectsq#
```

4. Cold start the region.

Migrating file and queue management (Oracle and SFS)

“Moving file management from SFS to Oracle” on page 54 and “Moving file management from Oracle to SFS” on page 56 describe how to change CICS queue and file management on an existing region. Review the following information before using these procedures:

- The CICS queue files do not need to be transferred if the region is to be cold started after changing the file manager, or if they are empty. The **cicssdt** and **cicsodt** commands can be used to list the number of records in each file.
- For mass-migration of a large number of files, see the README file that is in the directory for your operating system (see Table 12):

Table 12. The location of the README file on the AIX, HP-UX, and Solaris platforms

Platform	Directory
AIX	<code>/usr/lpp/cics/utls/cicsoram</code>
HP-UX	<code>/opt/cics/utls/cicsoram</code>
Solaris	<code>/opt/cics/utls/cicsoram</code>

- Although all CICS queues behave as fully recoverable when Oracle is used as the file manager, the recovery attributes, as defined in the Temporary Storage Definitions (TSD) and Transient Data Definitions (TDD) entries for the queues, are not changed. These settings determine which file is used to store data for

each queue. Therefore, when migrated from Oracle, queues that are defined as nonrecoverable will be nonrecoverable in SFS.

- All records that are over 4005 bytes are treated as variable by CICS using Oracle as the file manager.
- Migrated CICS applications that were written to access fixed records that are longer than 4005 bytes will find the following changes in CICS behavior:
 - On a READ, the value that is returned in the length field is the true length of the data, not the expected fixed length.
 - On a READ, the LENGTH option must be specified; otherwise, LENGERR is returned.
- When Oracle is used as the file manager, files that are over 4005 bytes long must be created with the last column of a variable type (raw, long raw). The total length of the preceding fixed columns cannot exceed 4005 bytes.
- The process of migrating CICS files from SFS to Oracle is subject to the following restrictions:
 - SFS files that have a fixed length of over 4005 bytes must be redefined.
 - SFS files that have a fixed length of over 4005 bytes and have key fields beyond byte 4004 cannot be directly migrated. The file must be redesigned to ensure that all key fields are within the first 4005 bytes of the record.
- File locking and concurrency on Oracle differs from file locking and concurrency on SFS. See the *TXSeries for Multiplatforms Application Programming Guide* for more information.

Moving file management from SFS to Oracle

The purpose of this procedure is to move CICS queue and file management from SFS to Oracle for a region that was configured to use an SFS server.

Notes:

1. This procedure creates Product Definitions (XAD) that define the Oracle database to the region. If you already have an XAD defined for this database, either delete it by using the **cicsdelete** command, or use the **-s** option with the **cicsoraconf** command. Otherwise, two XAD entries can possibly exist for this database and lead to unpredictable results.
2. DB2 does not allow particular special characters in the field names. Before migrating SFS files to Oracle, refer to the the Oracle SQL reference and see information that describes field name restrictions.

This procedure uses the **cicsoraconf** command to configure the region and the Oracle database for file management. To use Oracle explicit authentication, use the **-u** option to specify the user ID or password, or both. (Alternatively, you can use the CICSORAPASSWORD environment variable to specify the password.) If you want to specify the creation of a Product Definitions (XAD) entry for single-phase commit, use the **-n** option. If you do not want to create a Product Definitions (XAD) entry for this database (when one has already been created), use the **-s** option.

To use the **cicsoraconf** command to configure the region and the ORA database for file management:

- You must be logged in as **root** on the CICS server host, and **root** must be set up as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
- You must verify that the database has been started.

- If you are using explicit Oracle authentication to connect CICS to Oracle:
 - The user must be set up and the minimum permissions granted. (See the *TXSeries for Multiplatforms Installation Guide* for more information.)
 - If you do not want to be prompted for the password, you must set CICSORAPASSWORD environment variable to the database user ID password.

1. Shut down the region. Enter the command:

```
cicscp -v stop region regionName
```

2. Reconfigure the region for Oracle. Enter the command:

```
cicsoraconf -r regionName -C -u cics/cics
```

Note: If the database is residing remotely, you must configure a local net service, and use the following command to connect to the database. For example, if ora92 is the local net service for the remote database, give the command:

```
cicsoraconf -r regionName -C -u cics@ora92/cics
```

3. To transfer the files from SFS to Oracle:

a. Identify the files that the region requires:

- All files that are defined in the region's File Definitions (FD). The name of the file is specified with the FD **BaseName** attribute.
- CICS queues that are defined in the Region Definitions (RD). See Table 13.

Table 13. Queue name values (where %R is the region name)

RD attribute	SFS name (default)	Oracle name (mandatory)
LogicalTDQFile	%Rcicstdqlgfile	%Rlogtdq#
PhysicalTDQFile	%Rcicstdqphfile	%Rphtdq#
NonRecTDQFile	%Rcicstdqnofile	%Rnrectdq#
RecTSQFile	%Rcicsrectsqfil	%Rrectsq#
NonRecTSQFile	%Rcicsnrectsqfile	%Rnrectsq#
LocalQProtectFile	%Rcicsplqfile	%Rlqprot#
LocalQFile	%Rcicsnlqfile	%Rlque#

b. Transfer the files from SFS to flat file format. Use the **cicssdt** command with the **stof** option. For example:

```
cicssdt -s serverName -c stof BaseName
```

Specify N in the Dump file Asis? field.

```
[Flat Filename .....: BaseName.sdt
```

```
[Dump file Asis ? ..... Y/[N]: N
```

```
[Dump Record(s) .....: ALL
```

c. Create the user-defined files in Oracle.

d. Transfer the files from flat file format into Oracle. Use the **cicsodt** command with the **ftod** option. For example:

```
cicsodt -c ftod tableName -u userid/password
```

Note: If you do not want to be prompted for the userid and password, you must set CICS_XDT_USERPASS. For example:

```
CICS_XDT_USERPASS=userid/password
```

Use the same flat file name that you created in step 3b. You must type over the whole field. For example:

```
[Flat Filename .....: BaseName.sdt
```

For more information, see the *TXSeries for Multiplatforms Administration Reference*.

Note that the queue files must be imported under the Oracle names that are shown in Table 13 on page 55. For example, to import the recoverable transient data queue file for regionA, use the following commands (assuming that the SFS server file is using the default name):

- 1) Enter the command:

```
cicssdt -s ../cics/sfs/hostABCD -c stof regionAcicsrectsqfile
```

This command produces a flat file named **regionAcicsrectsqfile.sdt**.

- 2) Enter the command:

```
cicsodt -c ftod regionArectsq#
```

This command prompts you for the name of the flat file.

- 3) Enter:

```
regionAcicsrectsqfile.sdt
```

4. Cold start the region.

Moving file management from Oracle to SFS

This procedure moves CICS queue and file management from Oracle to SFS for a region that was configured to use Oracle.

Before using the procedure, do the following:

- Review “Migrating file and queue management (DB2 and SFS)” on page 49 so that you fully understand the effects of this change.
- Note that recovery attributes are ignored when Oracle is used for CICS queue and file management. Verify that the following attributes are set correctly for use with SFS:
 - The Transient Data Definitions (TDD) **RecoveryType** attribute
 - The Temporary Storage Definitions (TSD) **RecoverFlag** attribute
 - The File Definitions (FD) **RecoverStatus** attribute

If the attributes are not set correctly, delete the queues and redefine the attributes; File Definitions (FD) can be changed.

1. Use a normal shutdown to shut down the region. For example:

```
cicscp -v -l /tmp/cicscp.log stop region regionA
```

2. If an SFS server does not exist that this region can use, use the instructions that are given in “Overriding the defaults imposed by the cicscp start sfs_server command” on page 29 to set one up.
3. Transfer the files from Oracle to SFS:
 - a. Identify the files that the region requires:
 - All files that are defined in the region’s File Definitions (FD). The name of the file is specified with the FD **BaseName** attribute.
 - CICS queues that are defined in the Region Definitions (RD). See Table 13 on page 55.
 - b. Transfer the files from Oracle to flat file format. Use the **cicsodt** command with the **dtof** option. See the *TXSeries for Multiplatforms Administration Reference* for more information.
 - c. Create the user-defined files in SFS.
 - d. Use the **cicssfscnf** command to configure the SFS server. For example, if the region name is regionA and the SFS server is **../cics/sfs/hostABCD**, use this command:

```
cicssfsconf -R wc regionA DefaultFileServer=./cics/sfs/hostABCD
```

The **-R** option resets the file system to SFS in the Region Definitions (RD) and File Definitions (FD) and renames the queues. Use the **-I** option if the queue files were transferred in a previous step.

Attention: This deletes any system-defined Product Definitions (XAD) for this region.

- e. Transfer the files from flat file format into SFS. Use the **cicsodt** command with the **ftos** option. See the *TXSeries for Multiplatforms Administration Reference* for more information.

Note that the queue files must be imported under the SFS server names that are shown in Table 13 on page 55. For example, to import the recoverable TDQ file for regionA, use the following commands (assuming that the SFS server file is using the default name):

- 1) Enter the command:

```
cicsodt -c dtof regionArectsq#
```

This command produces a flat file named **regionArectsq#.ddt**.

- 2) Enter the command:

```
cicssdt -s ./cics/sfs/hostABCD -c ftos regionAcicsrectsqfile
```

This command prompts you for the name of the flat file.

- 3) Enter:

```
regionArectsq#
```

- 4. Cold start the region.

Chapter 5. Working with CICS queues

The queues that are needed by CICS to pass data and internal state information are added when the region is created. These queues include the Temporary Storage Queues (TSQ), Transient Data Queues (TDQ), and internal queues that CICS uses for auxiliary temporary storage queues and intrapartition data queues.

The file manager must be started before any CICS region that is to use it is started. The file manager oversees the administration of files that hold the user's data and of queues that pass data and information about the state of the CICS system.

"Creating the resource definitions for adding queues to the SFS server" describes how the intrapartition queues are enabled for the use of the SFS server.

"Creating the resource definitions for adding queues to the DB2 file manager" on page 62 discusses how intrapartition queues are enabled for the use of the DB2 server.

"Creating the resource definitions for adding queues to the Oracle file manager" on page 64 discusses how intrapartition queues are enabled for the use of Oracle.

"Setting up temporary storage queues (TSQ)" on page 66 and "Setting up transient data queues (TDQs)" on page 68 describe how to set the definitions that enable the use of queues with CICS.

The chapter assumes that you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide* to set up a default CICS system.

Creating the resource definitions for adding queues to the SFS server

This section provides an overview of resource definitions that are used for queue and file management on an SFS server. These resource definitions are automatically created and set to the default values when the **cicscp create region** or the **cicscp start region** command is issued. The following list identifies the classes of resource definitions that are automatically created when the region is created with the **cicscp** commands:

- Region Definitions (RD)
- Structured File Server Definitions (SSD)
- Temporary Storage Definitions (TSD)
- Transient Data Definitions (TDD)
- File Definitions (FD)
- Schema File Definitions (SCD)

You can use the CICS resource management commands **cicsadd**, **cicsupdate**, and **cicsdelete** to modify the default values for some attributes of these resource definitions. See Chapter 8, "Working with CICS resource definitions," on page 115 for examples of these commands.

The queues that the CICS file management system uses are created automatically when the **cicscp create region** and **cicscp start region** commands are issued.

You can also use the CICS resource management commands **cicsdefault** and **cicssfscnf** to create these queues. When the **cicsdefault -r region_name** command (or the **cicsimport** command) is issued, the **/var/cics_regions** directory is created. This directory holds the default Region Definitions (RD). See “Creating the resource definitions for adding queues to the SFS server” on page 59 for a list of these RD and a description of their purpose. The **cicsdefault** command must be run before the **cicssfscnf** command because the **cicssfscnf** command uses the RD that is created by the **cicsdefault** command.

When the **cicssfscnf** command is issued, it calls **sfsadmin** commands, which set up the Temporary Storage Definitions (TSD) and Transient Data Definitions (TDD) to enable the SFS server for queue management. If your application programs are designed to use auxiliary temporary storage queues or intrapartition transient data queues, define them as described in “Configuring the Temporary Storage Definitions (TSD)” on page 67 and “Configuring the Transient Data Definitions (TDD)” on page 69.

The following list provides information about how the SFS server uses the resource definitions that are automatically created when the **cicscp create region** or **cicscp start region** commands are used. Additionally, if you want to use the CICS resource management commands, instead of the **cicscp** commands, to create and start your region, you can find in this section the CICS resource management commands that can be used to create these resource definitions.

- **Region Definitions (RD)**

Configure the region to use a specific SFS server by naming the SFS server with the **DefaultFileServer** attribute. This attribute points to the Structured File Server Definitions (SSD) for the SFS server and is used in the SFS server configuration process.

For each type of information that is stored on the SFS server, a set of files, indexes, and archives are created and a unique name is assigned to each. The default values that are supplied by CICS use the region name as a prefix. For example, the default value for the file that is used for locally queued unprotected starts is **%Rcicsnlqfile**. If you change a name in the Region Definitions (RD), ensure that the new name does not clash with any other filename in the same SFS server.

The following default indexes are used:

For locally queued unprotected starts:

LocalQFile: The name of the file

LocalQIndex: The index

LocalQMaxRecs: The maximum number of records

LocalQPrePages: The number of pages to preallocate

LocalQVol: The volume name

For locally queued protected starts:

LocalQProtectFile: The name of the file

LocalQProtectIndex: The index

LocalQProtectMaxRecs: The maximum number of records

LocalQProtectPrePages: The number of pages to preallocate

LocalQProtectVol: The volume name

For logically recoverable transient data queues:

LogicalTDQFile: The name of the file

LogicalTDQIndex: The index

LogicalTDQMaxRecs: The maximum number of records
LogicalTDQPrePages: The number of pages to preallocate
LogicalTDQVol: The volume name

For nonrecoverable transient data queues:

NonRecTDQFile: The name of the file
NonRecTDQIndex: The index
NonRecTDQMaxRecs: The maximum number of records
NonRecTDQPrePages: The number of pages to preallocate
NonRecTDQVol: The volume name

For nonrecoverable temporary storage queues:

NonRecTSQFile: The name of the file
NonRecTSQIndex: The index
NonRecTSQMaxRecs: The maximum number of records
NonRecTSQPrePages: The number of pages to preallocate
NonRecTSQVol: The volume name

For physically recoverable transient data queues:

PhysicalTDQFile: The name of the file
PhysicalTDQIndex: The index
PhysicalTDQMaxRecs: The maximum number of records
PhysicalTDQPrePages: The number of pages to preallocate
PhysicalTDQVol: The volume name

For recoverable auxiliary temporary storage queues:

RecTSQFile: The name of the file
RecTSQIndex: The index
RecTSQMaxRecs: The maximum number of records for all
RecTSQPrePages: The number of pages to preallocate
RecTSQVol: The volume name

The full set of Region Definitions (RD) for the SFS server is described in the *TXSeries for Multiplatforms Administration Reference*. Review this information before you begin using the SFS server definition procedures.

- **Structured File Server Definitions (SSD)**

The purpose of the Structured File Server Definitions (SSD) is to store the static and dynamic information that the SFS server requires at startup. It includes both control and initialization information. The name (key) of the SSD represents the name of the SFS server.

The SSD is defined with the **cicscp** commands or with the **cicssfscree** command, which is a CICS resource management command.

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about using SMIT to define the SSD.

Consider the following issues when you use the **cicssfscree** command to define the SSD:

- The logical volumes and SFS user ID. These are created by using operating system commands before the **cicssfscree** command is used. These names are defined in the SSD and are provided with the command. If you do not provide the names, defaults are assumed.

- The frequency with which data is written to the log is specified in the SSD. This specification affects recoverability and performance.

The full set of Structured File Server Definitions (SSD) is described in the *TXSeries for Multiplatforms Administration Reference*. Review this information before you begin using the SFS server definition procedures.

- **Temporary Storage Definitions (TSD) and Transient Data Definitions (TDD)**

The **cicssfscnf** command sets up the SFS server for queue management. If your application programs use auxiliary temporary storage queues or intrapartition transient data queues, define them as described in “Configuring the Temporary Storage Definitions (TSD)” on page 67 and “Configuring the Transient Data Definitions (TDD)” on page 69.

- **File Definitions (FD) and Schema File Definitions (SCD)**

The FD (one for each data file) describe how the file is to be used by the application program. Although the FD are not used in the configuration process, they must exist in order for CICS to use the file. See “Configuring the File Definitions (FD) in the permanent and runtime databases” on page 135 for more information.

The Schema File Definitions (SCD) can be used with the **cicssfsimport** command to help you set up files on SFS. The SCD contains information about file types, field types and lengths, and indexes. The same SCD can be used to set up several files. The full set of Schema File Definitions (SCD) attributes is described in the *TXSeries for Multiplatforms Administration Reference*. Review this information before you begin using the SFS server definition procedures. See also “Configuring the File Definitions (FD) in the permanent and runtime databases” on page 135.

Note: The **cicssdt** command, which can also be used to set up files on the SFS server, does not require Schema File Definitions (SCD). See “Setting up the data files on SFS using the cicssdt diagnostic tool” on page 139.

Creating the resource definitions for adding queues to the DB2 file manager

The DB2 configuration procedures that are described in “Preparing for DB2 file and queue management” on page 37 and “Configuring resources and DB2 with cicsdb2conf” on page 63 set the following resource definition attributes:

- Region Definitions (RD)
 - **FileSystemType** is set to **DB2**.
 - **DefaultFileServer** is set to the name of the database that is used for queue and file management for the region.
 - **RDBMSInstance** is set to the name of the instance that is used for queue and file management for the region.
- Product Definitions (XAD)
 - **XAOpen** is set to the string that is used to connect to the database.
 - **SwitchLoadFile** is set to a string that points to an object file that contains the **xa_switch_t** structure definition and XA-support subroutines for DB2.

Note: When used as described, the procedures use a default switch load file. If required, you can use your own switch load file, as described in “Preparing for DB2 file and queue management” on page 37.

Configuring resources and DB2 with cicsdb2conf

The **cicsdb2conf** command configures resource definitions and adds to the DB2 database the tables that are used for CICS queue and file management.

Attention: The **cicsdb2conf** command creates Product Definitions (XAD) that define this database to the region. If you already have an XAD defined for this database, either delete it by using the **cicsdelete** command or use the **-s** option with the **cicsdb2conf** command. Otherwise, two XAD entries can exist for this database, with unpredictable results.

Note: If DB2 returns an XA resource manager error at runtime (which possibly indicates a configuration error), CICS adds the error to the **/var/cics_regions/regionName/console.nnnnnnn** file.

These tasks must be done before you issue the **cicsdb2conf** command:

- The region must be configured.
- DB2 must be set up for queue and file management, as described in “Preparing for DB2 file and queue management” on page 37.
- If you are using explicit DB2 authentication to connect CICS to DB2:
 - The user must be set up and the minimum permissions granted.
 - If you do not want to be prompted for the password, you must set the **CICSDB2CONF_CONNECT_USING** environment variable to the database user ID password.
- A DB2 table must not be redefined while its corresponding file is opened to CICS. Use the CEMT SET FILE command to close and disable the file before redefining the underlying table.

Use the **cicsdb2conf** command as described in the *TXSeries for Multiplatforms Administration Reference*, for example:

```
cicsdb2conf -r regionA -C -i INSTANCE -a ALIAS -u DB2USER
```

This command has the following effects:

- The **RDBMSInstance** attribute is set to **INSTANCE**, and the **DefaultFileServer** attribute is set to **ALIAS** in the Region Definitions (RD) for the region.
- An XAD entry is created, which specifies that CICS interoperates with this instance of DB2 through the XA-interface, using two-phase commit. To use single-phase commit optimization, use the **cicsdb2conf** command with the **-n** option.
- The XAD **SwitchLoadFile** attribute is set to **cicsxadb2**, which is the default switch load file.

Note: If you are using your own switch load file, use the **cicsdb2conf** command with the **-s** option to suppress the creation of a system-defined XAD entry. Then use **cicsadd** to create the XAD for this database (if one does not already exist) and set the **SwitchLoadFile** attribute to the name of your switch load file.

- The XAD **XAOpen** attribute is set to **ALIAS,DB2USER**.
- The **FileServer** attributes in all the File Definitions (FD) that are defined for the region are set to null.
- The DB2 authentication service is invoked to verify that the user DB2USER is allowed access to this database. If the **CICSDB2CONF_CONNECT_USING** environment variable is set to the correct password for DB2USER, DB2USER is

authenticated. If one of these environment variables is not set, you are prompted for the password. If the `-u` option is not used, the environment variable `CICSDB2CONF_CONNECT_USER` can be used to set the DB2 user going to access the database.

Note: You can further restrict access to the DB2 database by using the `cicsdb2conf -g` option as described in the *TXSeries for Multiplatforms Administration Reference*.

- A table is created in the DB2 database for each of the following:

Table 14. Table names for CICS queues in DB2

Type of queue	Table name
Recoverable auxiliary temporary storage queues	%Rreqtsq#
Nonrecoverable auxiliary temporary storage queues	%Rnreqtsq#
Logically recoverable transient data queues	%Rlogtdq#
Physically recoverable transient data queues	%Rphtdq#
Nonrecoverable transient data queues	%Rnreqtdq#
Locally queued protected STARTS	%Rlqprot#
Locally queued unprotected STARTS	%Rlque#

Because of DB2 restrictions, all queues are recoverable and protected.

Creating the resource definitions for adding queues to the Oracle file manager

The Oracle configuration procedures that are described in “Preparing for Oracle file and queue management” on page 47 and “Configuring resources and DB2 with `cicsdb2conf`” on page 63 set the following resource definition attributes:

- Region Definitions (RD)
 - **FileSystemType** is set to **ORACLE**.
 - **DefaultFileServer** is set to the name of the oracle instance that is used for queue and file management for the region that is picked up from the environment variable `ORACLE_SID`.
 - **RDBMSInstance** is set to the name of the instance that is used for queue and file management for the region that is picked up from the environment variable `ORACLE_SID`.
- Product Definitions (XAD)
 - **XAOpen** is set to the string that is used to connect to the database.
 - **SwitchLoadFile** is set to a string that points to an object file that contains the `xa_switch_t` structure definition and XA-support subroutines for Oracle.

Note: Users must use their own switch load files, as described in “Preparing for Oracle file and queue management” on page 47.

Configuring resources and Oracle with `cicsoraconf`

The `cicsoraconf` command configures resource definitions and adds to the Oracle database the tables that are used for CICS queue and file management.

Attention: The **cicsoraconf** command creates Product Definitions (XAD) that define this database to the region. If you already have an XAD defined for this database, either delete it by using the **cicsdelete** command or use the **-s** option with the **cicsoraconf** command. Otherwise, two XAD entries can exist for this database, with unpredictable results.

Note: If Oracle returns an XA resource manager error at runtime (which possibly indicates a configuration error), CICS adds the error to the **/var/cics_regions/regionName/console.nnnnnnn** file.

These tasks must be done before you issue the **cicsoraconf** command:

- The region must be configured.
- Oracle must be set up for queue and file management, as described in “Preparing for Oracle file and queue management” on page 47.
- If you are using explicit Oracle authentication to connect CICS to Oracle:
 - The user must be set up and the minimum permissions granted.
 - If you do not want to be prompted for the password, you must set the **CICSORAPASSWORD** environment variable to the database user ID password.
- An Oracle table must not be redefined while its corresponding file is opened to CICS. Use the **CEMT SET FILE** command to close and disable the file before redefining the underlying table.

Use the **cicsoraconf** command as described in the *TXSeries for Multiplatforms Administration Reference*, for example:

```
cicsoraconf -r regionA -C -u cics/cicas
```

This command has the following effects:

- The **RDBMSInstance** attribute is set to the value that is in the **ORACLE_SID** environment variable, and the **DefaultFileServer** attribute is set to **ORACLE** in the Region Definitions (RD) for the region.
- An XAD entry is created, specifying that CICS interoperates with this instance of Oracle through the XA-interface, using two-phase commit. To use single-phase commit optimization, use the **cicsoraconf** command with the **-n** option.
- The XAD **SwitchLoadFile** attribute is set to **oraclexa_sta**, which is the default switch load file.

Note: If you are using your own switch load file, use the **cicsoraconf** command with the **-s** option to suppress the creation of a system-defined XAD entry. Then use **cicsadd** to create the XAD for this database (if one does not already exist) and set the **SwitchLoadFile** attribute to the name of your switch load file.

You must ensure that the switch load file is appropriate, depending upon whether static or dynamic registration is being used. Also, you must follow the procedures that are given in “Prepare the Oracle resource manager” on page 156 to build the new **genclntsh** in case dynamic registration is being used.

- The XAD **XAOpen** attribute is set to **XAOpen=Oracle_XA+SesTm=30+LogDir=/tmp+Acc=P//**. This attribute must be changed to **XAOpen=Oracle_XA+SesTm=30+LogDir=/tmp+Acc=P/cics/cics**. If the Oracle database is residing in a remote server, you must configure a local net service name and use that name with the user. For example, if **ora92** is the

local net service, the XAOPEN string could be set to
XAOpen=Oracle_XA+SesTm=30+LogDir=/tmp+Acc=P/cics/ cics+sqlNet=ora92.

- The **FileServer** attributes in all the File Definitions (FD) that are defined for the region are set to null.
- The Oracle authentication service is invoked to verify that the user **cics** is allowed access to this database. If the CICSORAPASSWORD environment variable is set to the correct password for **cics**, user **cics** is authenticated. If one of these environment variables is not set, you are prompted for the password.

Note: You can further restrict access to the Oracledatabase by using the **cicsoraconf -g** option as described in the *TXSeries for Multiplatforms Administration Reference*.

- A table is created in the Oracle database for each of the following:

Table 15. Table names for CICS queues in Oracle

Type of queue	Table name
Recoverable auxiliary temporary storage queues	%Rreqtsq#
Nonrecoverable auxiliary temporary storage queues	%Rnreqtsq#
Logically recoverable transient data queues	%Rlogtdq#
Physically recoverable transient data queues	%Rphtdq#
Nonrecoverable transient data queues	%Rnreqtdq#
Locally queued protected STARTS	%Rlqprot#
Locally queued unprotected STARTS	%Rlque#

Because of Oracle restrictions, all queues are recoverable and protected.

Setting up temporary storage queues (TSQ)

When an application creates a temporary storage queue, CICS gives it the attributes that are defined in a Temporary Storage Definition (TSD) template.

Temporary storage queues that the region requires are set up when the region and file manager are set up. See information about the **cicscp** command in Chapter 3, “Configuring and modifying CICS regions,” on page 17.

The *TXSeries for Multiplatforms Application Programming Guide* describes how to use the CICS API to store data in TSQ, either in main storage, or in auxiliary storage on a direct-access storage device.

TSQ templates that are defined in the region database and in the Temporary Storage Definitions (TSD). Temporary Storage Definitions (TSD) determine the attributes that are given to TSQ for the region. For complete information on Temporary Storage Definitions (TSD) and their attributes, see the *TXSeries for Multiplatforms Administration Reference*.

About temporary storage queues (TSQ)

Data for the main TSQ is held in memory. Data for the auxiliary TSQ is held in files. These files are placed onto the file manager that is specified with the Region Definitions (RD) **DefaultFileServer** attribute when the file manager is set up. If

you use the procedures that are described in Chapter 3, “Configuring and modifying CICS regions,” on page 17, the files are given the required names, indices, and structure.

Table 16 lists the TSQ file and index names for a CICS region that is using an SFS server.

Table 16. Names of TSQ files and indices for a CICS region that is using an SFS server

Type of queue	SFS file name (default)	SFS index named (default)
Recoverable	<i>regionName</i> cicsrectsqfile	cicsrectsqidx
Nonrecoverable	<i>regionName</i> cicsnrectsqfile	cicsnrectsqidx

Table 17 lists the TSQ file and index names for a CICS region that is using DB2.

Table 17. Names of TSQ files and indices for a CICS region that is using DB2

Type of queue	DB2 file name (mandatory)	DB2 index name (mandatory)
Recoverable	<i>regionName</i> rectsq#	dx <i>regionName</i> rectsq#
Nonrecoverable	<i>regionName</i> nrectsq#	dx <i>regionName</i> nrectsq

Table 18 lists the TSQ file and index names for a CICS region that is using Oracle.

Table 18. Names of TSQ files and indices for a CICS region that is using Oracle

Type of queue	Oracle file name (mandatory)	Oracle index name (mandatory)
Recoverable	<i>regionName</i> rectsq#	dx <i>regionName</i> rectsq#
Nonrecoverable	<i>regionName</i> nrectsq#	dx <i>regionName</i> nrectsq

Configuring the Temporary Storage Definitions (TSD)

The following command examples show how to add, change, delete, or view the Temporary Storage Definitions (TSD) that are in the permanent database. Full descriptions of Temporary Storage Definitions (TSD) and their attributes, including default values, can be found in the *TXSeries for Multiplatforms Administration Reference*.

- Adding a new TSD:

This example shows how to add a TSD entry with a key of **QUE1**. The defaults are assumed for all other TSD attributes.

```
cicsadd -c tsd -r regionA -P QUE1
```

- Changing an existing TSD:

This example shows how to change a TSD entry so that it is not installed into the runtime the next time the region is cold started.

```
cicsupdate -c tsd -r regionA -P QUE1 ActivateOnStartup=no GroupName=''
```

To install the definition into the runtime, see “Installing a resource definition from the permanent database” on page 118.

Note: If the **GroupName** attribute contains a group name that is included in the region’s **Groups** list, the TSD entry is installed regardless of the **ActivateOnStartup** value.

- Deleting a TSD entry:

This example shows how to delete a TSD entry from the permanent database.

```
cicsdelete -c tsd -r regionA -P QUE1
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- Viewing a TSD entry:

This example shows how to view a TSD entry.

```
cicsget -c tsd -r regionA QUE1
```

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Setting up transient data queues (TDQs)

Transient Data Queues (TDQ) contain attributes that define to CICS the symbolic destination to or from which transient data is routed in the CICS region.

The name of every TDQ must be defined to CICS in the Transient Data Definitions (TDD) before the queue is used by an application program. Unlike temporary storage queues, TDQs cannot be dynamically created by an EXEC CICS WRITEQ command from an application.

For complete information on Transient Data Definitions (TDD) and their attributes, see the *TXSeries for Multiplatforms Administration Reference*.

About transient data queues (TDQs)

Transient data queues that the region requires are set up when the region and file manager are set up.

The *TXSeries for Multiplatforms Application Programming Guide* describes how to use the CICS Application Programming Interface (API) to queue data for subsequent internal or external processing. Selected data, specified in the application program, can be routed to or from predefined symbolic destinations, either *intrapartition* or *extrapartition*.

Destinations are intrapartition if associated with a facility that is allocated to the region; they are extrapartition if associated with an external destination. Extrapartition queues are ASCII files that are in the operating system’s file structure. The external destinations of these queues must be defined in the Transient Data Definitions (TDD). Extrapartition queues are not managed by the file manager; they are accessed through EXEC CICS APIs.

Each CICS region requires three files for TDQs: one holds logically recoverable TDQ data, one holds physically recoverable TDQ data, and one holds nonrecoverable TDQ data. These files are placed onto the file manager that is specified with the Region Definitions (RD) **DefaultFileServer** attribute when the file manager is set up. If you use the procedures that are described in Chapter 4, “Working with a file manager in a CICS region,” on page 23, the files are given the required names, indices, and structure.

Table 19 on page 69 lists the TDQ file and index names for a CICS region that is using an SFS server.

Table 19. Names of TDQ files and indices for a CICS region that is using an SFS server

Type of queue	SFS file name (default)	SFS index name (default)
Logically recoverable	<i>regionName</i> cicstdqlgfile	cicstdqlgidx
Physically recoverable	<i>regionName</i> cicstdqphfile	cicstdqphidx
Nonrecoverable	<i>regionName</i> cicstdqnofile	cicstdqnoidx

Table 20 lists the TDQ file and index names for a CICS region that is using DB2.

Table 20. Names of TDQ files and indices for a CICS region that is using DB2

Type of queue	DB2 file name (mandatory)	DB2 index name (mandatory)
Logically recoverable	<i>regionName</i> logtdq#	dx <i>regionName</i> logtdq#
Physically recoverable	<i>regionName</i> pgtdq#	dx <i>regionName</i> phtdq#
Nonrecoverable	<i>regionName</i> nreqtdq#	dx <i>regionName</i> nrectdq#

Table 21 lists the TDQ file and index names for a CICS region that is using Oracle.

Table 21. Names of TDQ files and indices for a CICS region that is using Oracle

Type of queue	Oracle file name (mandatory)	Oracle index name (mandatory)
Logically recoverable	<i>regionName</i> logtdq#	dx <i>regionName</i> logtdq#
Physically recoverable	<i>regionName</i> pgtdq#	dx <i>regionName</i> phtdq#
Nonrecoverable	<i>regionName</i> nreqtdq#	dx <i>regionName</i> nrectdq#

Configuring the Transient Data Definitions (TDD)

The following command examples show how to add, change, delete, or view the Transient Data Definitions (TDD) in the permanent database. Full descriptions of Transient Data Definitions (TDD) and their attributes, including default values, are given in the *TXSeries for Multiplatforms Administration Reference*.

- To add a new TDD:

This example shows how to add a TDD entry with a key of **QUE1**. The defaults are assumed for all other TDD attributes.

```
cicsadd -c tdd -r regionA -P QUE1
```

- To change an existing TDD:

This example shows how to change a TDD entry so that it is not installed into the runtime the next time the region is cold started.

```
cicsupdate -c tdd -r regionA -P QUE1 \
  ActivateOnStartup=no \
  GroupName=''
```

To install the definition into the runtime, see “Installing a resource definition from the permanent database” on page 118.

Note: If the **GroupName** attribute contains a group name that is included in the region’s **Groups** list, the TDD entry is installed regardless of the **ActivateOnStartup** value.

- To delete a TDD entry:

This example shows how to delete a TDD entry.

```
cicsdelete -c tdd -r regionA -P QUE1
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- To view a TDD entry:

This example shows how to view a TDD entry.

```
cicsget -c tdd -r regionA QUE1
```

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Working with TDD in the runtime database

To work with the Transient Data Definitions (TDD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change Transient Data Definitions (TDD) while the region is running. Any changes that are made by using CEMT apply only to the current session of CICS. They are replaced with the permanent resource definitions on the next restart. CEMT lets you:

- View the runtime Transient Data Definitions (TDD).
- View the queue type of a runtime TDQ.
- View or change the trigger level of a runtime TDQ.
- View the queue to which an indirect runtime TDQ is pointing.
- View the name of the transaction to be triggered.
- View the name of the terminal associated with the runtime TDQ.
- View the status of runtime TDQs.
- Change the runtime TDQ status.
- Empty extrapartition runtime TDQs.

For more information, see the *TXSeries for Multiplatforms Administration Reference*. To write CICS application programs to manage the status or operation of transient data queues in the runtime, see the *TXSeries for Multiplatforms Application Programming Reference*.

Chapter 6. Working with CICS clients

A CICS client provides an interface to CICS regions, which enables system administrators to perform region management and enables CICS users to request data from the region and submit data to it. End users of CICS connect as clients to CICS regions to use CICS application programs, services, and resources.

It is recommended that you set up a CICS client on each CICS machine. A CICS machine can run either the client only, or it can run CICS regions and file managers also.

A CICS client can be any of the following programs:

- A Telnet client that emulates an IBM 3270 Information Display System and connects to a CICS region through a Telnet server by running the **cicsteld** server process.
- The External Call Interface (ECI) and External Presentation Interface (EPI), application programming interfaces that are used to enable non-CICS applications in client systems to communicate with CICS server programs that are running on CICS regions
- An IBM CICS Universal Client, which is a separately installed product that enables a client to connect directly to a region.

The information in this chapter assumes that you have created and configured a CICS client as described in the *TXSeries for Multiplatforms Installation Guide*. The purpose of this chapter is to present additional methods for client communication and to explain the mechanics behind each of the available types of clients.

“Using the Telnet client (cicsteld)” explains the use of the Telnet (**cicsteld**) server with CICS.

“Behavior of cicsteld” on page 77 describes the different purposes for which the **cicsteld** command is used.

“Using the ECI and EPI applications for client access” on page 78 explains how CICS uses ECI and EPI programming interfaces to enable connection to a region.

“Using the IBM CICS Universal Client products” on page 78 describes how to set up access to the CICS region from IBM CICS Universal Client products.

“Setting up Terminal Definitions (WD)” on page 83 describes how to define terminals to your CICS system.

Using the Telnet client (cicsteld)

The CICS Telnet server process (**cicsteld**) allows clients that can interpret an IBM 3270 Information Display System to connect to a CICS region. Telnet servers are available on several operating systems.

Connections to CICS through 3270 Telnet programs use a non-CICS product as the terminal program. It is important that the product provides the level of support for data streams that CICS transactions require. Many 3270 Telnet terminals do not support all the attributes that can be specified in BMS. Key mappings to 3270 keys

are specific to the individual client. To use Telnet to connect to a CICS region, the Telnet terminal must provide the **tn3270** protocol. See the documentation for the Telnet terminal for information about customizing the keys.

Telnet terminals and the **cicsteld** server

The Telnet terminal resides on the client's system, receives input, and communicates with the Telnet server process, **cicsteld**, on the destination machine. The **cicsteld** server process then communicates with the CICS region.

An instance of the **cicsteld** Telnet server must be started for every connection between a Telnet terminal and a CICS region. This can be done either implicitly or explicitly:

- Implicitly, as a child of the **inetd** daemon (recommended for production systems). See "Starting the **cicsteld** server with the **inetd** daemon" on page 73.
- Explicitly, as an operating system process, for example, by using the **cicsteld** command from the operating system command line. This method is helpful for testing. See "Starting the **cicsteld** server from the command line" on page 76.

When the CICS Telnet server (**cicsteld**) is available, a user issues the **telnet** command on the client machine. This command enables the client to connect to the **cicsteld** server process that provides access to CICS. When a region is selected, **cicsteld** connects to it and requests a terminal autoinstall. The user can then run CICS transactions. If it cannot connect to the region, the **cicsteld** process returns a message to the client.

Figure 6 shows a CICS system that is using a Telnet session to enable communications between the client machine and the server machine.

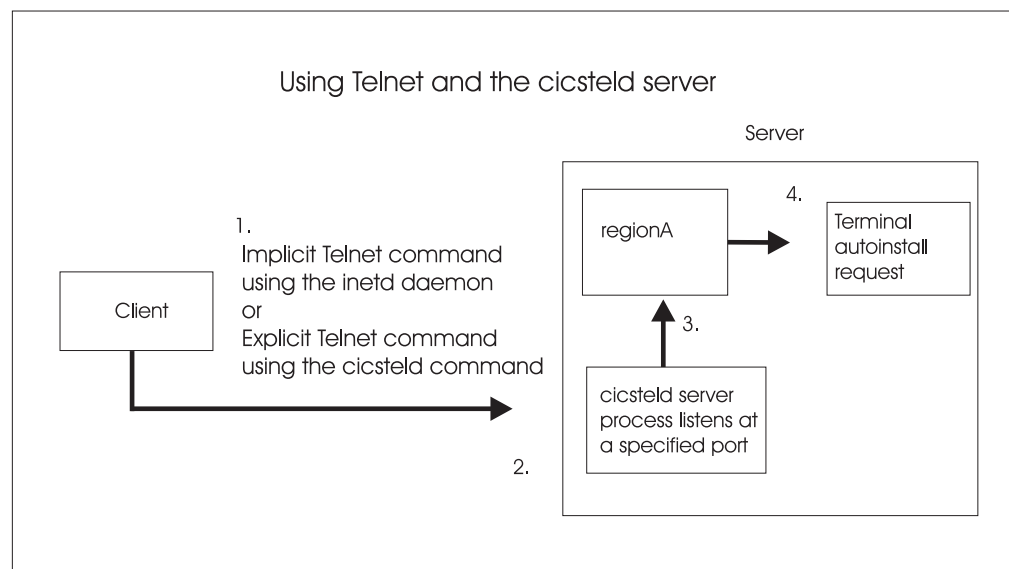


Figure 6. An example of a Telnet session in CICS

1. Area 1 in Figure 6 shows the two methods that can be used to initiate a Telnet session. The **inetd** daemon can be run. In addition, the **cicsteld** command can be issued from the command line. Both these methods start the **cicsteld** process on the server from within their programs. The **inetd** daemon is configured by using the **cicscp create telnet_server** command. See "Using the **cicscp** command to configure the **inetd** daemon" on page 73.

2. Area 2 shows a **cicsteld** process that is started on the server when the **cicsteld** command is issued. This process listens at a specified port for any client requests.
3. When the **cicsteld** process receives a client request, a connection is made to the region. This is shown in Area 3.
4. When a region receives a connection request from a client that is using the Telnet server process, the region issues a terminal autotinstall request. This is shown in Area 4. When the terminal is installed, a CICS session can be started, and CICS commands can be issued. See “Selecting a model type when the **cicsteld** server is used” on page 83 for more discussion on terminal selection.

Note: When the **cicsteld** server process receives an automatic transaction initiation (ATI) transaction, the keyboard remains locked if an ATI transaction is about to start. When the ATI transaction starts and issues a terminal request such as SEND, the keyboard is unlocked.

Starting the **cicsteld** server with the **inetd** daemon

The **inetd** daemon is an operating system process that runs on the server and listens for a client connection request. This daemon is started when the operating system is started. You can configure the **inetd** daemon to listen continuously on a specific port for attempts by clients to connect to the CICS region. Whenever the **inetd** daemon detects a connection attempt, it creates an instance of the **cicsteld** server process. The client connection request can come from any client that supports the 3270 protocol.

Using the **cicscp** command to configure the **inetd** daemon:

Use the **cicscp create telnet_server** command to create a Telnet server configuration that can be used with the **inetd** daemon. A shell script is automatically created in the **/var/cics_clients** file for each Telnet server that is created by the **cicscp** command. The program contains the arguments to the **cicsteld** command.

When the **cicscp create telnet_server** command is complete, check the **/etc/services** file for the port number that has been automatically assigned for the Telnet server. You must supply this port number to the configuration information for the **inetd** daemon.

The **-m** flag specifies a model definition on which to base the one that is to be created. It allows all parameter values except **NetName** (which must be unique) to be copied from the specified Telnet server. If the server is not created by the **cicscp** command, or if the shell script is edited manually after the server is created, the **cicscp** command can fail to retrieve the parameter values from the model server.

CICS provides an example shell script to aid in customizing the **inetd** daemon to work in your environment. Table 22 on page 74 shows the location of the sample script, the location where your customized script needs to be placed, and the variables that require modification for each platform.

Table 22. Locations for the *inetd* daemon scripts and variables requiring customization

Platform	Locations of scripts	Customized variables
AIX	Example script: <code>/usr/lpp/cics/etc/cicsteld.sh</code> Customized script: <code>/usr/lpp/cics/bin/cicsteld.sh</code>	NLSPATH LANG
HP-UX	Example script: <code>/opt/cics/etc/cicsteld.sh</code> Customized script: <code>/opt/cics/bin/cicsteld.sh</code>	NLSPATH LANG
Solaris	Example script: <code>/opt/cics/etc/cicsteld.sh</code> Customized script: <code>/opt/cics/bin/cicsteld.sh</code>	NLS_PATH LANG LD_LIBRARY_PATH

Use the following procedure to enable the **inetd** daemon to start the **cicsteld** server process. To configure the **inetd** daemon, you must be logged in as **root**.

1. Use the appropriate shell script to customize the **inetd**. Copy the sample file **cicsteld.sh** from the directory where it is located on your platform, to the customized location that is indicated for your platform in Table 22.
2. Ensure that the necessary environment variables are set correctly for your system. Table 22 shows which environment variables need to be customized on each platform.
3. The **inetd** daemon needs to know the port numbers on which it is to listen, the program to run when a connection request arrives, and any optional parameters to pass to the program; you provide this information by editing the **/etc/services** file and the **/etc/inetd.conf** file. See “Setting parameters in the **/etc/services** file” on page 75 and “Setting parameters in the **inetd** daemon configuration file” on page 75 for information about the required settings.
4. When you have added new entries to this file, the **inetd** daemon must read the new version.

On the AIX platform. Use the command:

```
refresh -s inetd
```

On the HP-UX platform. Use the command:

```
inetd -c
```

On the Solaris platform. Use the **ps** command to find the **inetd** process identifier, then enter:

```
kill -HUP pid
```

If the Network Information Server (NIS) is in use, the **/etc/services** file must be edited on the NIS master machine and the **ypmake** command must be executed to make the revised copy available to the client machines.

Setting parameters in the `/etc/services` file: The `/etc/services` file is a table that contains entries for the service name, the TCP/IP port number, and the type of TCP/IP connection that is to be used. The format of an entry is:

serviceName portNumber/protocol

where:

- *serviceName* is unique within the `/etc/services` file; no duplicate service names are allowed. It must match the service name that is specified in the `/etc/inetd.conf` file.
- *portNumber* is the port number on which the **inetd** daemon listens and through which the **inetd** daemon connects. The port number can be in the range 1 through 65535, and it must be unique.
- *protocol* can be tcp or udp. To connect to CICS, you must specify tcp.

Here is an example entry in the `/etc/services` file:

```
cicsteld 6192/tcp
payroll 6193/tcp
```

In this example, the Telnet process passes the `cicsteld` and `payroll` service names to the ports 6192 and 6193, using TCP as the communications protocol.

You must enter into this file the values that are appropriate to your system. These values are passed to the region when the **inetd** daemon is run.

Setting parameters in the `inetd` daemon configuration file: The `/etc/inetd.conf` file is the configuration file for the **inetd** daemon. It contains the following information:

.serviceName. .dataStream. .protocol. .wait. .userid. .program. .parameters.

where:

- *serviceName* must be unique within the `/etc/inetd.conf` file. It must match the service name that is in the `/etc/services` file.
- *dataStream* can be stream oriented or datagram oriented. For the **cicsteld** server, this must be stream.
- *protocol* can be tcp or udp. To connect to CICS, you must specify tcp.
- *wait* controls the serialization of connections that are allowed to the same port number. Specify `wait` to serialize requests or `nowait` to allow multiple concurrent requests.
- *userid* specifies the operating system user ID with which **cicsteld** server runs.
- *program* specifies the program that is to be run when a client connection request arrives. You must specify the full pathname of the program. Table 22 on page 74 shows the location of the program for each supported platform.
- *parameters* specifies **cicsteld** command parameters that are to be passed to the program that is specified in the *.program.* parameter. Specify the name of the executable file for the first parameter; that is, **cicsteld.sh**. See the *TXSeries for Multiplatforms Administration Reference* for more information about the **cicsteld** command parameters.

You must enter into this file the values that are appropriate to your system. These values are passed to the region when the **inetd** daemon is run.

On the AIX platform. For example, the following entry in the `/etc/inetd.conf` file starts a terminal that runs a payroll transaction on regionA:

```
payroll stream tcp nowait cics /usr/lpp/cics/bin/cicsteld.sh \
    cicsteld.sh -t pay1 -p principal -r regionA
```

In this example, `payroll` is the service name, and `pay1` is the CICS transaction that is to be run. `-t .initialTransaction.` can be used to specify a transaction that forces users to sign on (for example, CESN) to be run at initial connection after autoinstall.

The location of the `cicsteld.sh` script follows the directory structure that is shown in Table 22 on page 74. This example shows the use of the `cicsteld.sh` file in its customized location for the AIX platform.

Starting the cicsteld server from the command line

When the `cicsteld` Telnet server is started, the port that is allocated for attaching the client is specified in one of two ways:

- With the `-P` (port number) option on the `cicsteld` command.
- In the `/etc/services` database for the `cicsteld` server. The operating system can then query this file and attempt to allocate the port.

The `cicsteld` process then listens on that port for a single incoming request for attachment from a client.

Use the following procedure to start the `cicsteld` Telnet server from the operating system command line. Before you begin, ensure that the directory that contains the `cicsteld` program is in your path. Table 22 on page 74 shows the location of the `cicsteld` program for each supported platform.

1. Specify a valid port number. Do not use a port number that is specified in the `/etc/services` file.
2. Start the `cicsteld` server by issuing the command:

```
cicsteld -P portNumber -p principalName -r regionName &
```

- *portNumber* is the numeric port number that the `cicsteld` server listens on for incoming client requests to the Telnet server. If the port number is invalid, the `cicsteld` process issues an explanation message. If you do not specify a port number, the `cicsteld` process takes the port number from the service name entry of `cicsteld` in the `/etc/services` file.
- *regionName* is the region to which the `cicsteld` server connects. If a region is not specified, the Telnet client is presented a menu of available regions to which the user can connect.

If message ERZ018001E is displayed, saying that the selected port is already in use by another application, select a different port number.

Setting up the cicsteld server for use in DBCS or MBCS environments

CICS can be enabled to work with the `cicsteld` server process in environments that use double-byte character sets (DBCS) or multibyte character sets (MBCS).

You must run `cicsteld` command with the locale and model that are supplied by command options. The locale is specified by using the `-l` option. See Table 4 on page 12, Table 5 on page 12, and Table 6 on page 13 for a list of the locales that are available for each platform that can be supplied with the `-l` option. The model is specified by using the `-e` option. The following list shows the possible models that can be supplied with the `-e` option.

- `-e hft-mb`

- **-e mft-mb**
- **-e lft-mb**

On the AIX platform. For example, to run **cicsteld** on the AIX platform with a low-function, multibyte, monochrome terminal, and a ja_JP locale, a region called regionA and a port assignment of 9006, enter:

```
cicsteld -r regionA -P 9006 -l ja_JP -e lft-mb &
```

If the environment variable LANG is set, it is not necessary to specify the **-l** parameter.

Using Telnet to connect a client to a region

From the client, invoke the Telnet server and specify the host name and port number.

A panel is displayed that prompts you to select a region. Use **F7** and **F8** to scroll through the list of region selections. Select a region to which to connect, by moving the cursor to the relevant screen line and pressing **Enter**. (If you are already connected to a region that was predetermined when the Telnet server was set up, the region connection is made without the need to display the region selection panel.) After you select a region, **cicsteld** connects to the region and requests a terminal autoinstall.

To cancel the session instead of connecting to a region, press **F3**.

If you need to sign onto a region, use the CESN transaction. To log off from the CICS region, use the CSSF transaction.

Destroying a Telnet server configuration

The **cicscp destroy telnet_server** command destroys a Telnet server configuration.

See the *TXSeries for Multiplatforms Administration Reference* for more information.

Behavior of cicsteld

The **cicsteld** command acts as a 3270 Telnet server that allows connections to a CICS region by using a generic 3270 Telnet client. points to note about the behavior of a Telnet server (cicsteld) are:

- Telnet does not distinguish carriage return (**\r**) from newline (**\n**). If the keys on the terminal emit the carriage-return character, be sure to specify it as **\r** and not **\n** in the 3270 key-mapping file. The characters that are emitted by a particular key can be checked in the **terminfo** database. For example, the IBM 3151's capabilities are described in the file **/usr/share/lib/terminfo/ibm.ti**. In this file you can see that while the 3151 has unique capabilities, it shares most of them with the IBM 3161 device. Under **ibm3161**, **F1** is defined as generating "**\ea\r**", which is the escape character followed by a lowercase "a", followed by a carriage-return character.
- The **cicsteld** command converts data into the 3270 Telnet protocol and sends it to the Telnet client for display. It performs no terminal emulation.
- The **cicsteld** command communicates with a 3270 Telnet client by way of EBCDIC 3270 data because this is the format that the 3270 Telnet protocol requires. It cannot display lowercase single-byte characters in a multibyte locale because no such EBCDIC code page exists.

- The Entire 3270 data stream is sent in EBCDIC to the Telnet client from the Telnet server, but the final display of data is limited by the capabilities of the Telnet client that is being used.
- When **cicsteld** receives an ATI, and the transaction is scheduled, no form of 3270 data stream exists that the **cicsteld** command can send to lock the keyboard at the 3270 emulator. The keyboard remains unlocked until the new transaction starts.

Using the ECI and EPI applications for client access

The IBM CICS Universal Client product provides non-CICS application programming interfaces that enable external access to CICS facilities. Those interfaces are:

- The external call interface (ECI)
- The external presentation interface (EPI)

The following configuration is required for ECI and EPI applications to access the region:

- If the user ID that is passed with the connection request from the client is not defined to the region, or if no user ID is passed, the user is signed on with the region's default user ID. If this user ID does not have enough access privileges, **cicsadd -c ud** can be used to set up User Definitions (UD) for the passed user ID. This command is used on the machine that is running the region. See the *TXSeries for Multiplatforms Intercommunication Guide* for more information.

See the *TXSeries for Multiplatforms Intercommunication Guide* for more information.

Using the IBM CICS Universal Client products

IBM CICS Universal Client products allow workstation applications to communicate with the CICS External Presentation Interface (EPI) and External Call Interface (ECI) to enable transaction routing and distributed program link (DPL) calls to CICS servers. This section describes how to use IBM CICS Universal Client products with a CICS on Open Systems region. The IBM CICS Universal Client products are available on many platforms. Figure 7 on page 79 shows an example of an IBM CICS Universal Client communicating with a CICS region.

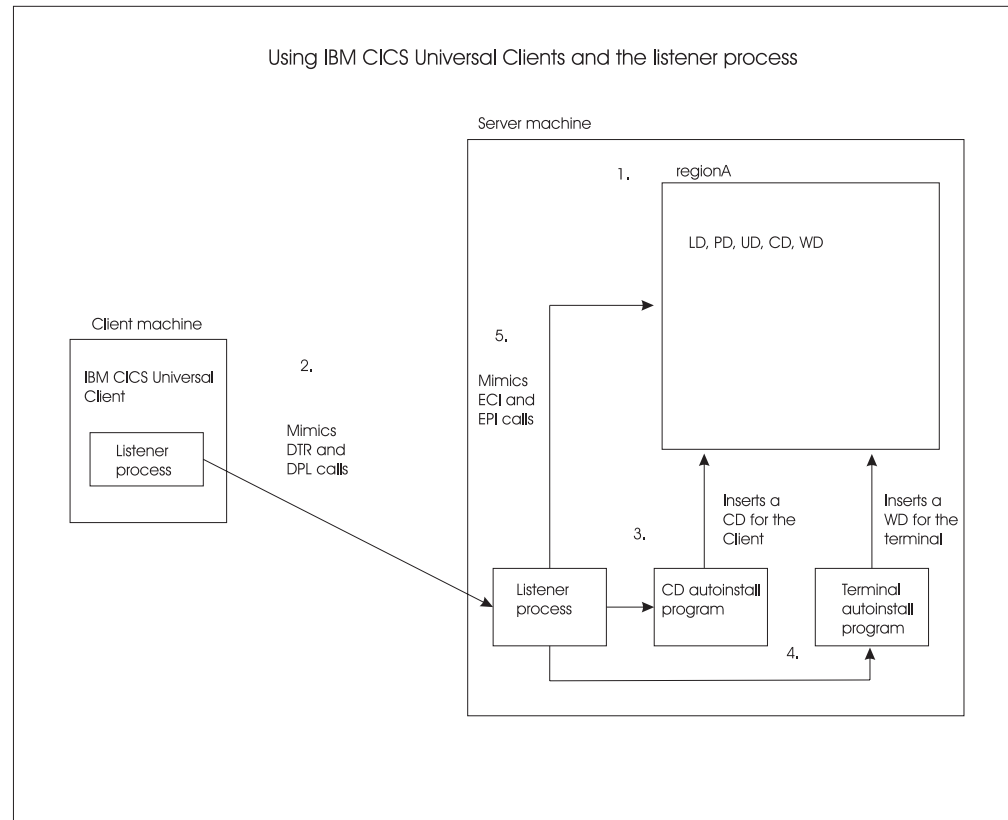


Figure 7. An example of an IBM CICS Universal Client session

When the IBM CICS Universal Client is used, a connection can be made either directly to an IBM mainframe platform, or it can be made through a TXSeries for Multiplatforms region. The numbered areas in Figure 7 identify processing that is associated with the using of an IBM CICS Universal Client to connect to a TXSeries for Multiplatforms region:

1. Area 1 shows the resource definitions that must exist in the region's stanza files for the IBM CICS Universal Client product that is to be used. The Listener Definition (LD) entry activates the listener process on the server. The listener process calls the Communications Definitions (CD) autoinstall program, which must have a Program Definition (PD) in the region's stanza files. A User Definition (UD) might be needed to ensure that enough privileges exist for the User ID that is passed with the client's connection request or for the region's default User ID, which is used if no ID is passed with the connection request.
2. Area 2 shows the communications between the listener process that is on the client machine and the listener process that is on the server machine, regardless of the server's platform. The IBM CICS Universal Client product imitates Dynamic Transaction Routing (DTR) calls and Distributed Program Links (DPL) to communicate between the two listener processes.
3. Area 3 shows that the listener process invokes the CD autoinstall program, for which a PD exists in the region. This program inserts a CD entry for the client into the region's stanza files.
4. Area 4 shows that the listener process also calls the terminal autoinstall program. The terminal autoinstall program installs a WD entry for the client's terminal.
5. Area 5 shows that the listener process imitates External Call Interface (ECI) or External Programming Interface calls to communicate with the region.

Using the Communication Definitions (CD) autoinstall program

Note: This section applies to autoinstall requests from IBM CICS Universal Client products. For more information about using CICS-supplied user programs and user exits, see Chapter 15, “Customizing CICS with user exits and user programs,” on page 185.

For any CICS on Open Systems region to communicate with IBM CICS Universal Client products, a Communications Definitions (CD) entry is required. This CD is automatically installed when an IBM CICS Universal Client first requests communications with the server. Any client installation request that come from an IBM CICS Universal Client product causes server to call the Communications Definitions (CD) entry autoinstall program (DFHCCINX) . This program automatically installs the CD in the region's CD stanza file.

Through the DFHCCINX program, the client can either request a particular name by which to be known, or the client can let the server construct a name. (If a particular name is to be assigned to the client, the name must match the value of the **RemoteLUName** attribute of the region's CD entry.) In either case, the DFHCCINX program allows the installation of the client to be canceled, or the name under which it is to be installed to be changed, or any other processing to happen, for example, recording client usage.

When the client finishes communicating with the server, it requests to be removed from the CD. The DFHCCINX program is called again at this stage, to allow processing such as updating statistics.

A Program Definitions (PD) entry of DFHCCINX must be made to the region's Program Definitions (PD) stanza file for this Communications Definition (CD) entry autoinstall program. CICS supplies a default sample program, **cics_client**, as source code. This sample program is written in the C programming language. CICS also supplies this file as an executable file. Table 23 shows the location of this default sample C program and executable file on each platform:

Table 23. Location of the default C source and executable code for DFHCCINX

Platform	Source and executable code
AIX	Source: <code>/usr/lpp/cics/src/samples/ccinx</code> Executable: <code>/usr/lpp/cics/bin</code>
HP-UX	Source: <code>/opt/cics/src/samples/ccinx</code> Executable: <code>/opt/cics/bin</code>
Solaris	Source: <code>/opt/cics/src/samples/ccinx</code> Executable: <code>/opt/cics/bin</code>

A makefile is also supplied in the sample directory, and a header file, **cics_client.h**, is supplied with the other header files.

The default sample program simply allows client installation to go ahead without making any changes. If you want to change the program, you can base your own user exit program on the sample code. Change the DFHCCINX PD **PathName** attribute to use your new program, following the normal rules for where programs

must be located. For more information, see “Using the ECI and EPI applications for client access” on page 78 and the *TXSeries for Multiplatforms Intercommunication Guide*.

Understanding how the listener process works

All CICS on Open Systems regions support requests from the IBM CICS Universal Client products by way of a *listener process*. This multithreaded process is started at region startup if a Listener Definitions (LD) entry exists with the **Protocol** attribute set to **TCP**. For information about how to configure the listener process, see “Configuring the region for IBM CICS Universal Client products” on page 82.

The listener process, which resides on the server, communicates with a corresponding process that resides on the client machine. The communications between these processes imitate the transaction-routing flows and the distributed program link (DPL) requests between CICS regions. The listener process uses an external presentation interface (EPI) call to imitate the flow that is generated between CICS regions when a transaction-routing operation is invoked. The flows that are generated by an external call interface (ECI) request are used by the listener process to imitate the CICS-to-CICS distributed program link (DPL) requests. The client process, therefore, appears to the server as a limited terminal-owning region (TOR). (More information on Listener Definitions (LD) can be found in the *TXSeries for Multiplatforms Intercommunication Guide* and the *TXSeries for Multiplatforms Administration Reference*.)

Because the data flows are the same as those between CICS systems, the server requires a Communications Definitions (CD) entry for the client system. When the client process is started on the client, flows that start the initialization transaction (CCIN) are sent to the server. This transaction takes care of the autoinstall of the client system in the CD. (See “Using the Communication Definitions (CD) autoinstall program” on page 80.)

It is not possible to hard-code CD entries that will be used by IBM CICS Universal Client product clients. However, if the client’s code page is a different from the default value of **RemoteCodePageTR** in the CD entry, the CD entry must be changed. This applies to all DBCS or MBCS environments.

Client definitions that must match server definitions

The client configuration is done by modifying the **CTG.INI** file. The following entries are of interest:

- **Protocol**: Set to **TCPIP**.
- **NetName**: The name of the server machine on the network. It is normally in the form *machine.company.com*, but can be the Internet address.
- **Port**: The port ID on which the server is listening. If you are using the default port (1435), this entry can be set to 0, which is the default. If the server is listening on another port number, substitute the value here.
- **UpperCaseSecurity**: Change this entry from the default **Y** to **N**. Many systems use mixed-case characters for user IDs and passwords, and specifying **Y** here can result in authentication failures.

Notes:

1. When writing ECI programs, you can specify the name of the first transaction that is run by the server after initialization (**eci_transid**). This transaction is, by

default, the CICS mirror transaction (CPMI). If your application specifies a different transaction, specify the DFHMIRS program as the back-end program to which the transaction points.

2. To support a DBCS/MBCS environment correctly, you must specify the **ModelTerm** parameter as one of either **hft-mb**, **mft-mb**, or **lft-mb**.

Configuration limitations from the listener process

An operating system process can have a maximum of 2000 open file descriptors at one time. This limit is not normally a problem, but for the listener process it can be. Each time a client system connects to the server system, a thread is started to handle the system, and a port is assigned. This port mapping uses a file descriptor. Therefore, a single listener process cannot handle more than approximately 1950 clients (allowing an extravagant 50 for other open files). To handle this limitation, you can:

- Increase the TCPProcessCount attribute.

This attribute of a TCP/IP listener definition determines how many separate operating system processes CICS uses to listen on the defined port and address. The threads that are created when systems connect are balanced across the available processes. So, for example, when this attribute is set to 2, two listener processes are created that share the connected systems. If 2000 client systems are connected to the defined port or address, approximately 1000 connections are handled in each of the listener processes. See the *TXSeries for Multiplatforms Administration Reference Guide* for more information about setting this attribute.

- Use a second adapter.

A second adapter provides greater bandwidth into a system and also a second Internet address. This allows you to configure another listener definition for this new address. The LD **TCPAddress** attribute is filled out with the specific address. The individual client systems must be separately configured to connect with one address or the other, to spread the number of systems that are connected to any single listener process.

- Use a second port.

This method has the advantage that the Internet address of the server machine remains the same, but the port address is different. If the cost of an additional adapter is not justified, or the system performance is not restricted by the adapter performance, this option can be used. The LD **TCPAddress** attribute remains blank, and port addresses are explicitly defined. The individual client systems must be configured to connect with one port or the other, to spread the number of systems that are connected to any single listener process.

Configuring the region for IBM CICS Universal Client products

If IBM CICS Universal Client products request services from the region, a Listener Definitions (LD) entry with the **Protocol** attribute set to **TCP** is required:

To configure Listener Definitions (LD) for IBM CICS Universal Client products:

1. Use the **cicsadd** command to add an LD entry. In the following example, an LD entry CLIENTS with the protocol of TCP is added to the permanent database for regionA. The defaults are assumed for all other LD attributes.

```
cicsadd -c ld -r regionA -P CLIENTS Protocol=TCP
```

2. If a nonstandard port is being used, add an entry to the **/etc/services** file.
3. Refresh the **inetd** daemon after adding this entry. See step 4 on page 74 for platform-specific instructions.

4. Restart the CICS system to start the listener process. The LD entry is initiated when the CICS system is restarted.

The following LD attributes are used for IBM CICS Universal Client products:

- **<Key>**: The identifier that is used when the system references the listener that is running to support this definition.
- **Protocol**: For IBM CICS Universal Client product support, this is set to **TCP**.
- **TCPAddress**: The Internet address of an adapter in the server machine (the machine that is running the listener process). It tells the listener process to listen on a particular adapter. The default listens for client systems on all network adapters.
- **TCPProcessCount**: The count of operating system processes across which CICS balances the connecting of client systems. The default of 1 creates a single listener process.
- **TCPService**: If left blank, the port that is used is the default port for CICS clients. IBM has registered the port **1435** with the Internet Assigned Number Authority (IANA) for this purpose. Therefore, if this option is left blank, CICS starts a listener process on the port 1435 at region startup.

If a name is entered for this attribute value, it must match an entry in the `/etc/services` file. For example, if you want the listener process to listen on port 6789, set this attribute to **cicsUser**. In the `/etc/services` file, you must have the following line:

```
cicsUser      6789/tcp      #CICS Client Listener
```

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 to use SMIT to define resources to the region.

For a complete description of all the attributes that can be used for the listener process, see the *TXSeries for Multiplatforms Administration Reference*.

Setting up Terminal Definitions (WD)

Terminal Definitions (WD) define the terminals that are available in the region. Although you are supplied with predefined terminal definitions for several terminal types, you can add, modify, or delete these as required. Complete descriptions of the Terminal Definitions (WD) and their attributes are given in the *TXSeries for Multiplatforms Administration Reference*.

The *TXSeries for Multiplatforms Application Programming Guide* describes how CICS handles data translation, transaction identification, synchronization of input and output operations, and the session control that is needed to read from, or write to, a terminal. This frees the application from the responsibility of controlling terminals.

Selecting a model type when the **cicsteld** server is used

Figure 8 on page 84 shows the process that the **cicsteld** Telnet client uses to select a terminal model type:

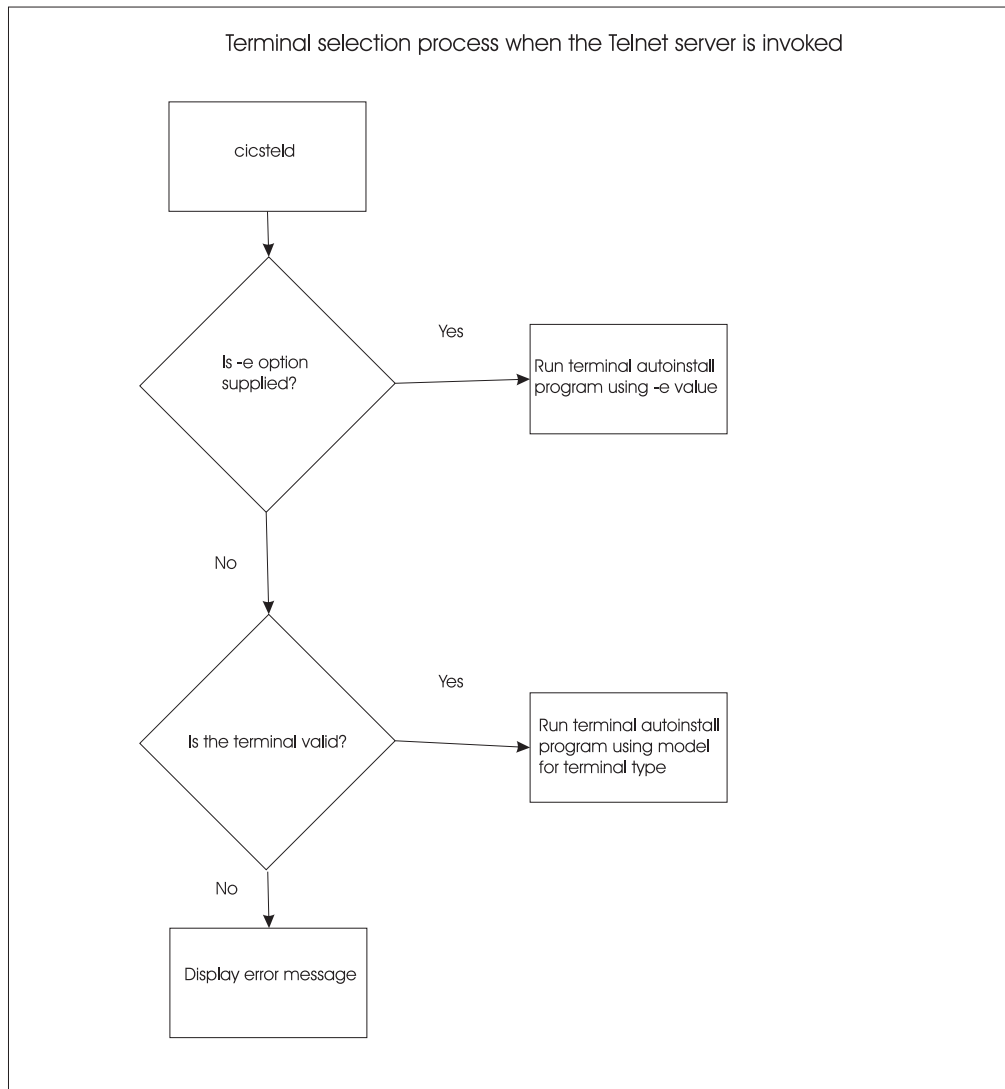


Figure 8. Terminal selection process for a client using Telnet server for CICS

The following process is initiated when a **cicsteld** Telnet server is invoked:

- The **cicsteld** command with the **-e** (emulation) option overrides the default model type that the **cicsteld** command selects automatically. This option allows you to specify a user-defined model type or a CICS-supplied model type. The preferred CICS model types that can be used for **cicsteld** are **hft** and **mft** for terminals with extended data streams, and **lft** for terminals that do not have extended data streams.
- When a region is connected for a client that is using Telnet, **cicsteld** checks whether the Telnet client is using a valid terminal. If the terminal is valid, **cicsteld** selects a model type for that terminal.

Selecting a model type when an IBM CICS Universal Client product is used

When an IBM CICS Universal Client product is initiating a connection request to the region, the behavior of the terminal autoinstall program is similar to that which

occurs when the **cicsteld** process is used. See “Selecting a model type when the cicsteld server is used” on page 83 for a description of the behavior of the terminal autoinstall program.

One exception exists in the behavior of the terminal autoinstall program when called by a connection request from an IBM CICS Universal Client product. “Terminal installation when an IBM CICS Universal Client uses an EPI application” discusses the conditions of this exception.

Terminal installation when an IBM CICS Universal Client uses an EPI application

When an IBM CICS Universal Client product EPI application connects to a region, CICS does a terminal autoinstall, which causes terminals of the format =xxx to be defined, where xxx is a character string that CICS chooses. This nomenclature is not the same as that used by the autoinstall of a CICS on Open Systems client, which generates a terminal ID that is based upon the first two characters of the client system. This can cause problems for those who have written their own autoinstall user exit programs and who did not expect the equal sign (=) character.

The CICS-supplied autoinstall models

When you invoke a **cicsteld** session, you define a model to be used as your autoinstall terminal to your CICS system. The autoinstall model that you use as a session model is identified by the value of the **DevType** attribute in the Terminal Definitions (WD) entry. The WD entries that are shipped with CICS contain several definitions that can be used as models. For more information about the autoinstall program, see the *TXSeries for Multiplatforms Administration Reference*. Table 24 lists each model.

Table 24. CICS-supplied autoinstall models

Key	Description
MDCL	For the IBM CICS Universal Client product model (device type ibm-cics-client)
MDCI	For the IBM CICS ECI model (device type ibm-cics-eci)
MD01	For the vt100 model (device type vt100)
MD02	For the aixterm model (device type aixterm)
MD03	For the aixterm-m model (device type aixterm-m)
MD04	For the ibm3151 model (device type ibm3151)
MD05	For the ibm3161 model (device type ibm3161)
MD06	For the ibm3164 model (device type ibm3164)
MD07	For the high-function multibyte color (device type hft-mb)
MD08	For the high-function single-byte color (device type hft)
MD09	For the high-function multibyte monochrome (device type mft-mb)
MD10	For the high-function single-byte monochrome (device type mft)
MD11	For the ASCII multibyte monochrome (device type lft-mb)
MD12	For the ASCII single-byte monochrome (device type lft)
MD13	For the hp terminal definition (device type hp)
MD14	For the hp term model definition device type hp term)
MD15	For the xterm model (device type xterm)
MD16	For the 327x-2-E for cicsteld (device type 3270C)

Table 24. CICS-supplied autoinstall models (continued)

Key	Description
MD17	For the 327x-2 for cicsteld (device type 3270M)
MD18	For the vt220 model (device type vt220)
MD19	For the dxterm (vt300) model (device type vt300)
MD20	For the dtterm model (device type dtterm)
MD21	For the 327x-2-E for cicsteld (device type 3270C-3)
MD22	For the 327x-2 for cicsteld (device type 3270M-3)
MD23	For the 327x-4-E for cicsteld (device type 3270C-4)
MD24	For the 327x-4 for cicsteld (device type 3270M-5)
MD25	For the 327x-5-E for cicsteld (device type 3270C-5)
MD26	For the 327x-5 for cicsteld (device type 3270M-5)
MD27	For the 97801 model (device type 97801)
MD28	For the Sun command tool and shell tool (device type sun-cmd)

Notes:

1. An autoinstall model is known by its **DevType** attribute.
2. To see the complete list of attributes for an autoinstall model, use the **cicsget** command as described in Appendix B, "Using CICS resource management commands for system management," on page 269.

Setting up and using terminals for DBCS or MBCS environments

You can use the following CICS-provided terminals in a double-byte character set (DBCS) or a multibyte character set (MBCS) environment:

- MD07: (hft-mb) High-function multibyte color
- MD09: (mft-mb) High-function multibyte monochrome
- MD011: (lft-mb) Low-function multibyte monochrome

Alternatively, you can create your own terminal definition. To define a terminal for a DBCS or MBCS environment, set its **SOSI** and **ProgramSymbol** attributes to **yes** in the Terminal Definitions (WD). You can do this by using the **cicsupdate** command.

For example, if the terminal key is **MINE**, and the region name is **regionA**, you can use the following example to update both the runtime and permanent Terminal Definition database:

```
cicsupdate -c wd -r regionA -B MINE SOSI=yes ProgramSymbol=yes
```

To set up a CICS Telnet server for a DBCS or MBCS environment, map the **sosiswitch** function to a key by adding the following statement to the **3270.keys** file:

```
bind
sosiswitch
"-e[013q"
shift-F1
```

Changing the definitions of a supplied model type

Use the **cicsupdate** command to change the attributes for a specific model type. For example:

```
cicsupdate -c wd -r regionName key \
          attribute=newValue \
          .
          .
          .
          attribute=newValue
```

where *regionName* is the name of the region, and *attribute=newValue* are the attribute and the value to which you want to change the attribute.

Administering Terminal Definitions (WD)

The following command examples show how to add, change, delete, or view the Terminal Definitions (WD) in the permanent database. Full descriptions of Terminal Definitions (WD) and their attributes, including default values, are given in the *TXSeries for Multiplatforms Administration Reference*.

- To add a new WD:

This example shows how to add a WD entry that is known to CICS as **TERM**; that is, a candidate for the autoinstall user exit program (see “Using the terminal autoinstall user program” on page 189) and is for terminals that support multibyte character sets. The defaults are assumed for all other WD attributes:

```
cicsadd -c wd -r regionA -P TERM ModelId=MyTerm SOSI=yes
```

If you want to add the definition to the runtime and permanent databases simultaneously, ensure that the region is running and use the **-B** (both permanent and runtime) option, as follows:

```
cicsadd -c wd -r regionA -B TERM ModelId=MyTerm SOSI=yes
```

- To change an existing WD:

This example shows how to change a WD entry so that it is not installed into the runtime the next time the region is cold started.

```
cicsupdate -c wd -r regionA -P TERM ActivateOnStartup=no GroupName=''
```

To install the definition into the runtime, see “Installing a resource definition from the permanent database” on page 118.

Note: If the **GroupName** attribute contains a group name that is included in the region’s **Groups** list, the WD entry is installed regardless of the **ActivateOnStartup** value.

- To delete a WD entry:

This example shows how to delete a WD entry.

```
cicsdelete -c wd -r regionA -P TERM
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- To view a WD entry:

This example shows how to view a WD entry.

```
cicsget -c wd -r regionA TERM
```

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Working with WD in the runtime database

To work with the Terminal Definitions (WD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change Terminal Definitions (WD) while the region is running. Any

changes that are made using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions on the next restart. Use CEMT to:

- View the runtime Terminal Definitions (WD)
- View or change the priority of a runtime terminal
- Put runtime terminals in and out of service in a CICS region
- Make runtime terminals available to automatic CICS transactions
- Make a runtime terminal unavailable to the transactions that it initiates
- Inquire about the network name of a CICS runtime terminal
- Purge tasks on CICS runtime terminals
- Inquire about the name of the autoinstall program
- Set the name of a new autoinstall program

For more information, see the *TXSeries for Multiplatforms Administration Reference*. To write CICS application programs to manage the status or operation of terminals in the runtime, see the *TXSeries for Multiplatforms Application Programming Reference*.

Chapter 7. Ensuring security in a CICS environment

CICS is made secure through the following facilities:

- Authentication of users who request a connection to the region
- Authorizing transaction and resource access through transaction level security (TSL) and resource level security (RSL)
- Enabling access to an External Security Manager (ESM)

This chapter describes how to implement authentication and TSL and RSL security. It also describes how to set permissions for file access with ACLs. It describes the basic concepts of CICS user security in a local environment. This includes security between the region (server) and terminal clients. See the *TXSeries for Multiplatforms Intercommunication Guide*; it builds on these concepts to describe how to use security facilities in an intersystem communications environment.

“Using an External Authentication Manager” on page 91 describes how to use an External Authentication Manager (EAM).

“Authorizing access to CICS transactions” on page 94 and “Authorizing user access to CICS resources” on page 99 describes how access to CICS transactions and CICS resources is restricted.

“Using an External Security Manager” on page 106 describes how to use the CICS-supplied security program, External Security Manager, within your CICS system.

Authenticating user access to CICS

CICS uses authentication to check the identity of a user, for example, by asking for a password. CICS allows access to a region only after the user is authenticated.

Users can be authenticated with a CICS user ID and password. If the **AuthenticationService** attribute in the Region Definitions (RD) is set to **CICS**, CICS security procedures are used for authentication. CICS provides resource definition attributes that can set authentication levels for RPCs between CICS objects.

Overview of CICS user authentication

To use CICS procedures for authenticating users, the following settings are required in the Region Definitions:

- The **AuthenticationService** attribute must be set to **CICS**.
- The **RuntimeProtection** attribute must be set to **none**.

When these attributes are set, CICS user authentication is performed by checking the credentials in User Definitions (UD stanza file).

To use CICS procedures for authentication by using EAM (External Authentication Manager), the following RD settings are required:

- The **AuthenticationService** attribute must be set to **CICS**.
- The **RuntimeProtection** attribute must be set to **none**.

- The EAMLoad attribute must be set to **yes**.
- The EAMModule attribute must be set to the External Authentication Manager Module name that the user has written.

In this case, the CICS user credentials are authenticated by an external source such as Windows Active Directory or LDAP.

For more information about EAM, see “Using an External Authentication Manager” on page 91.

Normally, **cicslterm** clients and Telnet servers are logged in as the region’s default user. If the user does not want to operate with the region’s default user ID, the transaction CESN must be used to provide a different CICS user ID, which can require a password, depending on how the user was defined.

CESN can also be used to change the password. In that case, the password is encrypted and stored in the User Definitions (UD), or in an external source if EAM is set. Note that if CESN fails, the user still has access as the default user ID.

You can use the **cicslterm** and **cicsteld** commands with the **-t** option to start the processes and invoke CESN in one step.

Security considerations in an XA-enabled environment

In an XA-enabled environment, access from CICS to a relational database is set at the region level. All transactions that run against an XA-enabled database use the same database user ID for access to that database.

This behavior is a result of the XA architecture, under which RM (Resource Manager) authentication occurs at the point where an **xa_open** call is issued. For CICS, the **xa_open** call is issued during the initialization of an application server. At this point, a connection is established to the relational database with a specific database user ID.

Each transaction that runs against this particular application server inherits the established database connection and, by implication, the established database authorization. Also, each CICS application server in a particular region connects to the database with the same database user ID (because it uses the same **xa_open** string). Therefore, every transaction that makes EXEC SQL calls, makes them under the authority of this specific database user ID.

In an XA-enabled CICS environment, base your security policy around CICS security facilities. CICS has facilities to control user access to the system and user access to specific resources; for example, the transactions that are available to users. This is described in “Authorizing user access to CICS resources” on page 99.

With DB2, you can *bind* a program (or package) to a database. For CICS transactions that use static SQL statements, the binder’s privileges, instead of the caller’s privileges, are used to access the database. Because all transactions are executed under the same database user ID, this user ID must still be granted EXECUTE privileges against packages that involve CICS transactions. CICS transactions that use dynamic SQL have their access authentication done at runtime, in the same way as other supported RDBMS systems do. For further information about DB2 security in an XA environment, see the *DB2: Administration Guide*.

Using an External Authentication Manager

An administrator can use an External Authentication Manager (EAM) as an alternative to the local CICS User Definition Authentication that uses UD.stanza. EAM is a user-supplied program that allows CICS users to authenticate by using the CICS user ID and password that are defined in an external source such as Windows Active Directory, RACF®, LDAP, or any other authentication source. CICS provides a sample EAM module to demonstrate the interface to which you need to conform if you choose to write your own EAM.

This EAM module is called whenever:

- A user ID and password combination needs authentication
- A password needs changing in the external user ID and password repository
- A user definition that is in UD.stanza is not present for the user who is trying to log on

After a successful password validation of an EAM user, EAM is called to install the user definition at CICS runtime.

How to use an EAM with CICS

By default, CICS uses internal authentication that uses UD stanza. To use an External Authentication Manager instead of CICS, you must:

1. Install the EAM module
2. Change the Region Definitions (RD) EAMLoad attribute to **yes**
3. Use the RD EAMModule attribute to specify the EAM program path and name

When the CICS region comes up, the EAMModule that the CICS Administrator specified is loaded into each cicsas process. When a CICS user tries to login with a user ID and password, CICS checks whether EAM is loaded. If the EAM is loaded, it passes that user ID and password to the EAM program for authentication.

Defining the interface between CICS and an EAM

You can install an EAM module that CICS can invoke automatically when it needs to authenticate a CICS user ID and password. CICS supplies an example implementation of this module to demonstrate the interface to which you need to conform when writing your own EAM. For information about this supplied module, see the *TXSeries for Multiplatforms Administration Reference*.

The EAM is a user-written module that implements four functions in its interface to the CICS module:

- Initialization
- Authenticate
- Change password
- User install

If the EAM module is required, the **load** subroutine loads it into the CICS system at CICS startup. The initialization function for an EAM is **CICS_EAM_Init**.

The four function names are used for reference only. The CICS system accesses these functions from the EAM program indirectly by address only. It does not access them from a call that names the function. Therefore, you can define the actual names that are used in the EAM program when you implement an alternative EAM. These functions are described in the following list:

- **Initialization** function: The **Initialization** function is called by the CICS startup process to enable the EAM to perform any system-specific initialization, and to determine the addresses of the EAM callback functions that perform authentication, change password, and user install.

This function is the entry point for the EAM module. The **load** subroutine call returns the address of the Initialization function to CICS, so that it does not need to be linked into CICS. CICS supplies the Initialization function with data that describes the CICS region. The EAM program returns, as data, the addresses of the three callback functions that perform authentication, password change, and user install. The following parameters are passed to the CICS_EAM_Init() Initialization function:

CICS_EAM_RegionData

Contains the system identifier of the CICS region. It contains also the CICS release version number.

CICS_EAM_FnBuf

Returns the addresses of the callback functions that perform authentication, password change, and user install.

- **Authenticate** function: The **Authenticate** callback function is called by CICS to perform validation of a CICS user ID and password. CICS calls the Authenticate() function when a user logs in. The function is passed with a CICS_EAM_RegionData structure and a CICS_EAM_Logon structure. The CICS_EAM_Logon structure contains the user ID and password that are to be authenticated.

Two parameters are passed to the function:

CICS_EAM_RegionData

Contains the system identifier of the CICS region. It contains also the CICS release version number.

CICS_EAM_Logon

Contains the user ID and password that the CICS user supplied when logging on.

The return codes from this function can be:

```
CICS_EAM_RETURN_OK
CICS_EAM_RETURN_WRONG_PASSWORD
CICS_EAM_RETURN_PASSWORD_EXPIRED
CICS_EAM_RETURN_BAD_NEWPASSWORD
CICS_EAM_RETURN_NO_SUCH_USER
CICS_EAM_RETURN_NOT_INITIALIZED
CICS_EAM_RETURN_USERID_REVOKED
CICS_EAM_RETURN_NOT_ACTIVE
CICS_EAM_RETURN_NOT_RESPONDING
CICS_EAM_RETURN_TRY_LOCAL
CICS_EAM_RETURN_INVALID
```

Parallel authentication: You can use the CICS local authentication, using UD.stanza, and the EAM both at the same time in the region. This action is known as parallel authentication. If you use both for authentication, preference is given to the EAM; that is, the user ID is validated firstly by the EAM. If the user ID is not present in the external authentication source (in the case of failure in the EAM authentication), and if the EAM has been written to return a code of CICS_EAM_RETURN_TRY_LOCAL, CICS checks for the user ID in UD definitions. If the user ID is not present in UD stanza, user authentication fails. If it does fail, or if the EAM does not tell the region to continue checking, the region uses the default userid permissions.

- **ChangePassword** function: The **ChangePassword** callback function is called by CICS to change the password for a particular user.

When a user requests to change the password, the old password is first validated. If the old password is validated successfully, the new password is accepted for change. This function is passed with a `CICS_EAM_RegionData` structure and a `CICS_EAM_ChangePassword` structure. The `CICS_EAM_ChangePassword` contains the user ID, the new password, and the old password.

The following parameters are passed to the **ChangePassword()** function:

CICS_EAM_RegionData

Contains the system identification of the region and the CICS version release number

CICS_EAM_ChangePassword

Contains the user ID, the old password, and the new password.

The return codes from this function can be:

```
CICS_EAM_RETURN_OK
CICS_EAM_RETURN_WRONG_PASSWORD
CICS_EAM_RETURN_PASSWORD_EXPIRED
CICS_EAM_RETURN_BAD_NEWPASSWORD
CICS_EAM_RETURN_NO_SUCH_USER
CICS_EAM_RETURN_NOT_INITIALIZED
CICS_EAM_RETURN_USERID_REVOKED
CICS_EAM_RETURN_NOT_ACTIVE
CICS_EAM_RETURN_NOT_RESPONDING
CICS_EAM_RETURN_TRY_LOCAL
CICS_EAM_RETURN_INVALID
```

`CICS_EAM_RETURN_INVALID` can be returned if the new password is not a valid password; for example, if it is too short, or does not meet password restrictions.

- **UserInstall** function: The **UserInstall** callback function is called by CICS to perform a User Definition autoinstall at run time. After a user has been authenticated, and if the user is not already defined to CICS, the function `UserInstall()` is called. Two structures are passed to this function: a `CICS_EAM_RegionData` and a `CICS_EAM_NewUser`. The structure `CICS_EAM_NewUser` contains the properties of the user that needs to be installed. The updated structure is passed back to the CICS region. The region gets the properties of the new user, and installs them in runtime. The structure `CICS_EAM_NewUser` has been defined in `cics_esm.h`, which is in the `/usr/lpp/cics/include` directory on AIX, and in the `/opt/cics/include` directory on Solaris and HP-UX.

The following parameters are passed to the **UserInstall()** function:

CICS_EAM_RegionData

Contains the system identification of the region and the CICS version release number.

CICS_EAM_NewUser

Contains ID, `TSLKeyList`, `RSLKeyList`, `Priority`, and `OpID` attributes, where:

ID Is the User ID that is to be installed.

TSLKeyList

Is the list of transaction security keys for this user.

RSLKeyList

Is the list of resource security keys for this user.

Priority

Is the user priority.

OpID Is the operator ID; for example, AIX for a user on AIX.

For more information about these attributes, see the *TXSeries for Multiplatforms Administration Reference*.

The return codes from this function can be:

CICS_EAM_RETURN_OK.

These structure definitions for all the callback functions are in the `cics_esm.h` header file. The sample EAM program is in the `/usr/lpp/cics/src/samples/eam` directory on AIX, and in the `/opt/cics/src/samples/eam` directory on Solaris and HP-UX.

Defining an EAM at startup

During initialization of each application server, CICS checks the value of the `EAMLoad` attribute in the Region Definitions (RD). If `EAMLoad` is set to **yes**, CICS loads the EAM module into the application server storage area. The region fails to start if the EAM fails to load. If the return code of the EAM program indicates a failure, CICS terminates the application server and the transaction.

The attribute `EAMModule`, which is defined in the RD, is used as an argument to the load call. `EAMModule` is either an absolute pathname to the EAM module, or it is a relative name. If a relative pathname is used, the `PATH` environment variable must contain the directory that contains the EAM module.

The `load call()` returns the entry point address of the EAM Initialization function. You must define this entry point address when you build the EAM module. The CICS authentication module calls the Initialization function of the EAM, to allow the EAM to perform any specific initialization. The EAM Initialization function uses a return code to indicate the success of the initialization. If an EAM fails to install, the startup process does not complete and the region shuts down.

If `EAMLoad` is set to **no**, any transactions or resources that have been defined as requiring external authentication default to using CICS internal user ID and password authentication.

Authorizing access to CICS transactions

This section describes the security mechanisms that CICS provides to restrict access to specified transactions.

Accessing user-defined transactions

CICS provides two methods that determine whether users or printer terminals are permitted access to specific transactions:

- Running a user-supplied External Security Manager (ESM) program
- Running internal CICS checking

You can use the Transaction Definitions (TD) **TSLCheck** attribute to define the method that is to be used.

When the **TSL** attribute is set to **external**, CICS passes security checking over to an ESM. If the RD attribute **ESMLoad** is set to **no**, security for the transaction defaults to CICS internal checking. This also applies to transactions that are defined during the life of the region.

For details of the ESM, see “Using an External Security Manager” on page 106.

When the **TSLCheck** attribute is set to **internal**, CICS internal security checking uses the TD **TSLKey** attribute and the User Definitions (UD) and Terminal Definitions (WD) **TSLKeyList** attributes to check whether users and terminals are permitted access to a transaction.

The TD **TSLKey** attribute specifies a value 1 through 64. The UD and WD **TSLKeyList** attribute is assigned a list of values 1 through 64. The **TSLKeyList** attribute must have the same value defined to it as is defined for the **TSLKey** attribute in order for the user or terminal to access the transaction.

For example, if **TSLKey** is set to 4 for the transaction and **TSLKeyList** is set to 3|5|6 for the user, the user is denied access to the transaction.

The Terminal Definitions (WD) **TSLKeyList** attribute is used when a terminal is a printer that is managed by the CICS Universal Client print program (**cicsprnt**).

You can define a transaction as having public access (**TSLKey** value of **public** or 1). In this case, the values **public** and 1 are assumed for every attempt to access the transaction, whether or not any of the following conditions apply:

- 1 is included in the **TSLKeyList**.
- A UD entry exists for the user.
- A user ID or terminal is associated with the transaction.

Transaction security for printer terminals

CICS allows you to associate TSL keys and RSL keys with a Terminal Definition (TD). This means that transactions can run at terminals where no user has signed on. This is used for printers that are attached to a **cicsprnt** terminal (output only).

You can define a printer terminal as an output-only device. You start the printer terminal by running a **cicsprnt** command that includes the network name of the terminal, the name of the region, and any printer options.

After the **cicsprnt** has been started, transactions can be directed to run at this terminal either by Automatic Transaction Initiation (ATI), such as explicit EXEC CICS START TRANSID commands, or by trigger TDQs. To decide whether the transaction can start, the system compares the **TSLKey** value for the transaction with the **TSLKeyList** value for the terminal. If the transaction's **TSLKey** value is in the terminal's list, the transaction can be started on the printer terminal.

Assume that you have a transaction PAYR that has a **TSLKey** of 15. Assume also that you have a printer attached to a terminal with a terminal ID of LPT1, which has a **TSLKeyList** value of 1, 8, 15, and 23. If you enter the following command, a security check is performed, and because the terminal has the correct security key, the transaction runs:

```
EXEC CICS START TRANSID(PAYR) TERMID(LPT1)
```

Require terminal keys only when no user sign-on is possible; therefore, define terminal security keys only for printer-attached terminals.

Security for automatic transaction initiation

You can specify security for transactions that are automatically initiated to a terminal by a transient data queue or by the CICS interval control program. (See the *TXSeries for Multiplatforms Intercommunication Guide* for more information.) Check the following to ensure that you can start all automatically initiated transactions without security violations:

- If the transaction is to run at a terminal with a signed-on user, ensure that the user at that terminal has the correct value defined for the **TSLKey** in the UD to run the transaction.
- If the transaction is to run at a terminal with no signed-on user, for example, at a printer, ensure that the terminal has the correct value defined for the **TSLKey** in the Terminal Definitions to run the transaction.
- If the transaction is not to be run at a terminal, note that nonterminal ATIs can have user IDs associated with them. Security keys are taken from the user ID if possible. Otherwise, only public transactions and resources are available. The **TSLKeyList** of the user who initiated the START must contain the security key of the transaction that is to be started.

Note: The START API checks both the TSL and RSL key of the transaction to ensure that the START can be issued, not that the transaction itself can start. See “Transactions that use EXEC CICS START or EXEC CICS CANCEL” on page 104 for more information.

For a transaction that is initiated by a transient data queue trigger, the user who issued the write operation that triggered the queue must have the correct TSL key to run this transaction.

An example of running a secure transaction

Assume that a user’s definition contains TSL keys **1**, **2**, and **4**. Two of the transactions in the user’s system are ACCO and PERS. ACCO has a TSL key of **2**, and PERS has a TSL key of **5**.

If the user invokes the ACCO transaction, CICS compares the TSL key of the transaction ACCO with the keys that are in the user’s TSL Key List. Because ACCO’s TSL key of **2** is in the list, the user is authorized to run the transaction.

If the user tries to invoke the PERS transaction, CICS compares PERS’s TSL key of **5** with the keys in the user’s TSL key list. Because CICS does not find a TSL key in the user’s list to match that of the PERS transaction, the following occur:

- CICS does not run the transaction.
- The transaction PERS abends with abend code A146 (security violation).
- CICS sends message ERZ014016 with the appropriate details to the terminal.
- CICS writes the error messages ERZ014016 and ERZ014001 to the **console.nnnnnn** log. This lets the operator who is checking the log know which user caused the violation.
- CICS writes the error message ERZ045004E to the CSMT log. This specifies a transaction Security violation and again specifies the user and terminal that attempted to run PERS.

Authorizing access to CICS-supplied transactions

CICS supplies many transactions to assist control and usability. The misuse of these transactions can seriously affect your operations.

The CICS-supplied transactions are initially defined with a standard transaction security value of **1** or **public** in the Transaction Definitions (TD). This simplifies the initial installation of CICS but has the disadvantage that it allows all CICS users to invoke these transactions.

CICS sign-on functions

Apart from running CRTE to route a transaction to another system, (thereby signing on there), the user requires only CESN to change his or her authority. For example, by default, all users can run at the same level of authority, but a user can occasionally need to use a different ID to execute a secure transaction.

CESN and the sign-off transaction CESF are by default defined with a TSL key of 1. Because most users are likely to use these transactions, the default security settings for these two transactions are normally not changed.

Security for system transactions

Transactions such as CEMT (Master Terminal), CECI (Command Interpreter), and CEDF (Execution Diagnostic Facility) perform some very powerful functions. Consider applying security to these transactions and requiring specific sign-on to privileged users to run them.

For example, programmers can use the CECS transactions to check command syntax interactively. They can also use the CECI transaction to execute commands without having to write a program. Because it is so easy to change data in your system with this facility, you will probably want to control the use of CECI.

In a production system, you can provide CEDF and CECI but allow their use only with appropriate control and supervision. Use both transaction-level and resource-level security checking to control them tightly, and reserve them for the system programmer to use when extreme difficulties are encountered and when debugging cannot be done any other way.

Authorizing access to the CEDF debugging transaction

You can use the CICS-supplied transaction CEDF to debug another transaction if the transaction has been compiled in debug mode (with the command **cicstran -e**).

If the CICS system has given you access to the CEDF transaction, CICS performs Transaction Security Level (TSL) validation to verify that you have enough authorization to debug the transaction. Also, if required, subsequent Resource Security Level (RSL) validation is performed for both the user who is running the transaction and for you as the CEDF user who is debugging the transaction.

CICS performs TSL checking slightly differently when validating a CEDF user who is debugging a transaction. The CEDF transaction and the debugged transaction require valid TSL security values; therefore, you must ensure that both the CEDF transaction and the transaction that you are debugging have the same values for the **TSLCheck** attribute (**internal** or **external**) in the Transaction Definitions (TD).

If CEDF requires resource validation (that is, the **RSLCheck** attribute in the TD is not equal to **none**), you must ensure that both CEDF and the debugged transaction have the same **RSLCheck** values (**internal**, **external**, or **none**) in the TD. If both these requirements are met, you can also run the transaction from the terminal on which the CEDF is being run.

If you are using CEDF and security checking is unsuccessful, the following message is displayed on your screen:

Security violation: Security Check does not allow debug access

Note: If you are routing transactions, the link security keys are used when the **TSLCheck** is performed for your user ID.

CICS performs RSL validation for you, in the same way as for the transaction user, by using your user identifier, terminal identifier, and link identifier (if routing transactions). Similarly, for an external check, CICS can pass information to the External Security Manager (ESM). That information describes your user identifier, terminal, and, if appropriate, the link from which you are routing transactions.

Managing transaction security

The following considerations and actions are necessary for managing transaction security:

- Decide about the degree of transaction security. Must it apply to every transaction, to each group of transactions, or just to selected systems?
The more intricate your transaction security becomes, the more difficult it will be to administer and maintain. Balance the need for protection with the need for usability.
- Assign transaction security keys (**1** through **64**) to the transaction identifiers that are involved.
- Check your transaction definitions (user transactions and CICS-supplied transactions). Change the TSL keys to the correct value if needed.
- If you are using external security for transactions, ensure that the definitions for the transactions and resources that use external security have their **RSLCheck** attributes set to **external**.
- Create your list of CICS users. You need the following:
 - Names of the users
 - Unique user IDs
 - Associated passwords
 - Appropriate lists of security keys to match the tasks that the user is to access
- Create a procedure for updating user information, for example, by making a form that specifies the following:
 - The requestor (name, phone number, location)
 - What is to be changed (tasks to be added or removed, password to be reset)
 - Sign-on user ID and password (security administration fields only)
 - Management approval
- Provide the following information to users:
 - The user ID and password
 - Which transactions they will be able to use
 - When the security will be implemented (if not already done)
 - Whom to contact in case of difficulties, such as the security administrator
- If you have any printers attached to CICS, and their attached terminal has a **TSLKeyList** or a **RSLKeyList** value that allows access to secure transactions, ensure that the printer is in a secure area.
- Be sure to monitor the relevant logs for attempted security violation messages, and determine the causes. See the `/var/cics_regions/regionName/console.nnnnnnn` and the `/var/cics_regions/regionName/data/CSMT.out` files.
- If moving from a test system to a production system, check which users, if any, require access to CEMT, CEDF, and CECL. Ensure that nobody can access these transactions except those who have a critical need to use them.
- Check each user's sign-on to ensure that the users can sign on with their user IDs and passwords and use the transactions and resources that they are permitted to use.

Authorizing user access to CICS resources

Resource Security Level (RSL) validation provides security for access from application programs to resources during the runtime.

RSL validation also applies to CICS facilities and is based on a concept that is similar to transaction security. For information about transaction security, see “Accessing user-defined transactions” on page 94.

Access from outside the runtime

The following CICS facilities are available from the operating system outside a running region:

- The CICS translator **cicstran**
- The Basic Mapping Support (BMS) processor **cicsmap**
- The resource definition files
- The interfacing command for Telnet servers: **cicsteld**
- The following CICS resource management commands: **cicsadd**, **cicsdelete**, **cicsupdate**, and **cicsget**

Note: When the region uses CICS password checking (that is, if the **AuthenticationService** attribute in the Region Definitions (RD) is set to **CICS**), the encrypted CICS password is stored in the User Definitions (UD). You can use the **cicsupdate** and **cicsadd** commands to change the CICS passwords. Any user who is in the group **cics** can change the passwords.

Authorization of access to all these facilities is controlled by the operating system instead of by CICS. Depending on what level of security you require on your site, you can, when you install resources, change the permissions that CICS gives to those resources.

Authorizing access to startup and shutdown programs

You must be logged in as **root** to use any of the startup or shutdown methods. The methods follow:

- **On all platforms.** The CICS control program (**cicscp**) commands
- **On the AIX platform.** The **startsrc** and **stopsrc** commands and the System Management Interface Tool (SMIT)
- **On the HP-UX or the Solaris platforms.** The **cicsstart** and **cicsstop** commands
- **On the HP-UX platform.** The System Administration Manager (SAM)

Authorizing access to resource definitions with CICS commands

Several of the CICS resource management commands are used to update resource definitions in the permanent and runtime databases (as described in Chapter 4, “Working with a file manager in a CICS region,” on page 23). You can set the permissions for your files and directories to prevent unauthorized users from invoking the individual commands; for example, the **cicsupdate** command.

On the AIX platform. You can perform the online definition of CICS resources by invoking the System Management Interface Tool (SMIT) to issue the CICS resource management commands. You can use the AIX permissions to allow only authorized users to invoke SMIT.

Because you possibly need to restrict the use of some SMIT options to particular users, this method can be unsatisfactory. In this case, use the AIX permissions on the individual commands. Note that if a user is using SMIT to invoke an unauthorized command, the user can respond to the prompts in the interface up to the point of invoking the actual command. At this stage, the command is not performed because of the security violation.

Use the permissions of your operating system to control which users can perform operations on which files, and which operations the users can perform. Some examples follow:

- The system administrator can be allowed to read, add, update, and delete records.
- Members of particular groups can be allowed to read and update resource definition files.
- All users can be allowed to read resource definition files.

To prevent unauthorized updates to the runtime definition of the resources, set up transaction security for the CICS-private transaction CURD, which performs this task. For further information, see “Authorizing access to CICS transactions” on page 94.

Authorizing access to resources outside the running region

CICS gives the following permissions to objects that it uses:

Table 25. Required permissions for CICS product files

Type of file	Permissions	Group
Resource definition files	rw-rw----	cics
Runtime executable files	r-x-----	cics
Files created and used by the runtime	r----- or rw----- as required	cics
Nonruntime executable files, except for cicstran , cicsmap , cicstcl , cicslterm , and cicsteld	r-xr-x---	cics
cicstran , cicsmap , and cicstcl	r-xr-xr-x	cics
cicslterm and cicsteld	r-xr-xr-x	cicslterm

Authorizing access to CICS resources from applications

CICS uses security keys to perform security checks that determine whether you have permission to access a given resource. You can specify for each resource whether or not CICS performs these checks on individual transactions by setting the **RSLCheck** attribute in the TD entry for the resource.

The RSL security checks that CICS makes can either be internal checks that are performed by CICS (**RSLCheck** set to **internal**), or external checks that are performed by a user-supplied External Security Manager (ESM) program (**RSLCheck** set to **external**).

Resource Security Level (RSL) validation can be performed on the following resource types:

- Transactions
- Programs
- Files

- Transient data queues
- Temporary storage queues
- Journals

You can also specify that RSL validation is not performed for particular transactions. Do this by setting the **RSLCheck** attribute in the required TD entries to **none**.

The security check that is performed for a specific resource is defined by comparing the value of the **RSLCheck** attribute in the TD with the value of the particular **RSLCheck** attribute for the resource specified in the Region Definitions (RD). The following attributes are defined in the RD:

- **TransactionRSLCheck**
- **ProgramRSLCheck**
- **FileRSLCheck**
- **TransientDataRSLCheck**
- **TemporaryStorageRSLCheck**
- **JournalRSLCheck**

The ESM performs resource level validation only if both the resource type in the RD and the transaction indicate that external validation is required.

If you have specified **external** for the **RSLCheck** attribute in the TD entry for the transaction and specified **yes** for the **ESMLoad** attribute in the RD, and the load of the ESM fails, the region fails to start up. If you have specified **external** for the **RSLCheck** attribute in the TD entry for the transaction and set **ESMLoad** to **no**, so that no ESM program exists, a message is displayed when the region starts up. It warns that the ESM is not loaded. Resource checking for the region defaults to internal checking. This action also applies to transactions that are defined during the life of the region. For more information, see “Using an External Security Manager” on page 106.

If the return code of the ESM program indicates failure to give access to the resource, CICS terminates the application server and the transaction.

If you require internal security checking, the CICS security module implements a validation scheme. An **RSLKey** attribute is included in the definitions that are provided for the specific resource. You can assign one of the following **RSLKey** values to each resource:

- A key that is represented by a number (1 through 24)
- **public**
- **private**

The User Definitions (UD) entry that is associated with a user’s identifier includes an **RSLKeyList** attribute. This attribute defines which resources the user is allowed to access. You can specify up to 24 keys in your resource security key list in the UD.

When a user attempts to access a specific resource, CICS obtains from the runtime version of the appropriate resource definition the **RSLKey** that is associated with that resource. If the key is present in the user’s **RSLKeyList**, CICS allows the user to access the resource. If the key is not present, CICS prevents the user from accessing the resource and raises an error against the request.

A similar **RSLKeyList** is included in the Terminal Definitions (WD) to enable security checking where no user identifier is associated with a transaction.

CICS uses terminal keys in specific conditions; for example, if the terminal is a printer that is managed by a **cicsprnt** process that an authorized user starts.

In all other cases, CICS uses the user keys.

Specific special **RSLKey** values in the appropriate resource definitions are used by CICS to indicate that security checks always succeed. You can define resources with the **RSLKey** value **public**, which grants access to any user. Users do not need to have any keys in their **RSLKeyList** attribute to access a **public** resource. Setting the **RSLKey** attribute to **private** prevents access by any user, and ensures that the resource can be accessed only from trusted transactions that have the TD attribute **RSLCheck** set to **none** in their TD entry.

Users cannot directly access a resource that is defined as **private**. To access a **private** resource through a trusted transaction (**RSLCheck** set to **none**), users need the relevant **TSLKey** or **RSLKey** to use that transaction, and the transaction can then update the **private** resource for them.

Using the TSLKeyList and RSLKeyList attributes

The **TSLKeyList** and **RSLKeyList** values against which transactions and resources are checked by CICS in a standalone CICS system are established from either the User Definitions (UD) or the Terminal Definitions (WD).

When a user successfully establishes a CICS session, CICS obtains the security keys that are associated with the user's identifier from the UD. CICS uses these keys to perform security validation for transactions and resources that are associated with the user's terminal device. However, some transactions are attached to devices to which the user cannot sign on; for example, printers. Although devices do not have entries in the UD, you can still provide transaction and resource security by defining transaction and resource security keys in the WD instead. Use terminal keys only where sign-on *cannot* be performed; for example, for a printer device that is executing transactions. CICS uses these keys when it is performing security validation for these devices.

More about resource security

The resource security level (RSL) is a second level of CICS security checking that extends the transaction security concept to resources such as files, transient data queues, and application programs.

To associate security with resources in the system, specify whether you want RSL checking to occur for a given transaction. To do this, establish an RSL key for each resource so that when a user enters a transaction identifier, the user's keys apply to any resource that are accessed by that transaction.

To define resource security:

1. Specify that you want RSL checking for the transaction.
2. Define an RSL value (or **public** or **private**) to each resource that the transaction accesses.
3. Restrict the RSL keys for users and terminals to the key values that are associated with the resources that the user accesses.

Resource security for the transaction

First define that you want RSL checking to occur whenever the transaction is run. Do this by setting an appropriate value in the **RSLCheck** attribute of the

Transaction Definitions (TD) for the transaction. The **RSLCheck** attribute has three possible states: **none**, **internal**, or **external**.

To use the internal security RSL keys to protect your resources, ensure that the **RSLCheck** attribute for the transaction is set to **internal**. (To use an External Security Manager to check your resources, set the **RSLCheck** attribute for the transaction to **external**.)

For supplied transactions (such as CECI, CPMI, CEDF, CEBR, CSM1 and CSMI), **RSLCheck** is set to **internal** by default. Therefore, all resources that are accessed during a command interpreter, execution diagnostic, or intercommunications support session, must have appropriate RSL keys defined. If this specification is not acceptable, you can change the definitions of these transactions, and specify the **RSLCheck** value of **none**. You will then be relying on transaction security to control the use of these transactions.

If a transaction is not associated with a terminal (for example, an ATI task that is running in the background), but it is associated with a user, the resource security checking is associated with the user. Do not use resource security checking for a transaction that is not associated with a terminal or a user.

Resource security for resources

After you have specified RSL checking for the transaction, the next step is to associate the appropriate RSL key with each resource that the transaction accesses. The **RSLKey** attribute associates RSL checking with the following resources:

- Files as defined in the File Definitions (FD)
- Transient data queues as defined in the Transient Data Definitions (TDD)
- Temporary storage queues as defined in the Temporary Storage Definitions (TSD)
- Journals as defined in the Journal Definitions (JD)
- Application programs as defined in the Program Definitions (PD)
- Transactions as defined in the Transaction Definitions (TD)

Note: Resources that are held by relational database managers (RDBMs) are not subject to RSL checking and must be protected by the security mechanisms that are provided with the RDBM.

Each resource can have only one RSL key assigned to it. The same key must be contained in the user's **RSLKeyList** to allow the user access to that resource.

See Table 26 for the types of access that are allowed for the various values of the **RSLCheck** and **RSLKey** attributes. If **RSLCheck** is set to **external**, the **RSLKey** value becomes irrelevant. A nonterminal transaction with an associated user ID (such as a function-shipping mirror) behaves in the same way as does a terminal-attached transaction for security checking. A nonterminal transaction without a user ID can access only public resources.

Table 26. RSL key values

RSL key value	RSLCheck=none (no checking)	RSLCheck=internal (checking)
RSLKey=private	Access allowed	Access refused
RSLKey=public	Access allowed	Access allowed
RSLKey=1–24	Access allowed	Access allowed (if keys match)

An **RSLKey** value of **private** is a special value that indicates that the resource can be accessed only by transactions that have an **RSLCheck** value of **none**.

If a user tries to access a resource and the resource security key does not match, a NOTAUTH condition is raised. You can provide your own code in your application programs to handle the NOTAUTH condition. For example, you can display a message that explains that the user is not authorized to access that resource. Message ERZ045001I is written to the CSMT log for the system administrator. It states that an unauthorized access of a resource was attempted. (See the *TXSeries for Multiplatforms Application Programming Guide* for more information.)

Files, transient data, and temporary storage

In many installations, the most important resources are the files of data that are used. The internal security **RSLKey** attribute gives you an opportunity to control access to those files.

For each file that is defined in the FD, specify an **RSLKey** value **1** through **24** in the UD or WD of those users who need to access that file. Provided that any transaction that is accessing the file has the **RSLCheck** attribute set to **internal**, that file is protected.

Transient data queues are used for queueing data for further processing. These queues can also contain sensitive data. You can define an **RSLKey** value for these intrapartition and extrapartition transient data queues in their TDD definitions.

Temporary storage can contain sensitive data, so protect these queues by setting an appropriate **RSLKey** value in their definitions in the Temporary Storage Definitions (TSD).

When you create a temporary storage queue from a template in the TSD, the temporary storage queue inherits the protection settings from the template.

Application programs

You can also consider application programs as resources. Remember that you associate each transaction definition with an application program to start the processing of the data for that transaction. However, a program can communicate with other programs by using the LINK and XCTL commands.

Applying resource security to CICS application programs ensures that their use is controlled in an appropriate way. A program can execute a LINK or XCTL command only to other programs that have an RSL key that matches a value in the user's **RSLKeyList**. You can define an RSL key for each program in the Program Definitions (PD).

Transactions that use EXEC CICS START or EXEC CICS CANCEL

This applies both for time-oriented initiation (EXEC CICS START) and for cancelation (EXEC CICS CANCEL) requests.

When you want a transaction to perform resource security checking, you must set its **RSLCheck** value to **internal**. This means that the transaction denies access to those resources with RSL keys that do not match a value for the user (or for the terminal if no associated user exists).

Now assume that one of those resources is another transaction. CICS performs resource-level security checking of references to a transaction only when an

application program issues a START or CANCEL command for another transaction identifier. An RSL check is made against the RSL key of the target transaction. The RSL check is made at the time the START or CANCEL command is issued for a transaction.

For example, assume that you have a transaction PERS that runs a program **persprg**. Transaction PERS has the **RSLCheck** attribute defined as **internal**. This means that when transaction PERS is running (which means that the program **persprg** is running), resource security checking is done on the resources that PERS accesses. When program **persprg** issues a START for a different transaction (transaction PAYR for instance), the **RSLCheck** attribute for the PAYR transaction is irrelevant at this point. Because PERS has a setting of **internal** for **RSLCheck**, PAYR must have an appropriate RSL key to be executed. It must have **RSLKey** set to either **public** or *nn*, where *nn* is a number 1 through 24 that matches a key in the UD **RSLKeyList** for the user who is running PERS.

Only after a successful RSL check does CICS attempt to run the transaction. Then, the TSL check is performed. After PAYR is started, the **RSLCheck** attribute in the TD for PAYR determines whether CICS also applies resource security checking when PAYR accesses resources.

After you have specified **RSLCheck** as **internal** for a transaction, you must have appropriate **RSLKey** values on all resources that the transaction uses. If any of those resources has an **RSLKey** of **private**, all attempts to access the resource are denied.

Resource security for journals

As with the other resources, journals can be subject to resource security checking, so you can set the **RSLKey** value for each journal.

Ensure that users have appropriate levels of operating system security, because journals reside on the file system as flat files.

Resource security for users

After you have set up all your transactions and resources for resource security, the final step is to establish the various RSL keys that give access to the transactions and resources. You specify the keys that each user has by defining them in the **RSLKeyList** attribute in the UD.

Unlike transaction security keys, an **RSLKEY** of 1 is not the same as public access. If a resource is defined with an **RSLKey** of **public**, no RSL key is needed in a user's **RSLKeyList**. However, if a resource has an **RSLKey** of 1, a user needs an **RSLKey** of 1 to access it.

Remember that if a resource has an **RSLKey** of **private**, it does not matter what **RSLKey** the user has. Access depends on the **RSLCheck** value. If a user is to have a key list with all the possible keys 1 through 24, you can set the list to **all** instead of using the setting 1|2|3...|24.

Resource security for terminals

Terminals that are attached to printers need to have appropriate **TSLKey** and **RSLKey** values defined so that when a secure transaction is executed and sends output to a printer terminal, the terminal, which has no attached user, has the authority to run it. The RSL keys give the terminal the authority to access the necessary resources as required by the started transaction.

The TSL and RSL terminal keys can be set in the Terminal Definitions. As with other **TSLKeyList** and **RSLKeyList**, the TSL keys can be 1 through 64, and the RSL keys can be 1 through 24.

Resource security checklist

Planning the implementation of RSL checking is more involved than planning for securing transactions. The resources that you want to make secure (for example, a file in the FD) can be accessed by many tasks. Consider implementing RSL checking in addition to transaction checking.

1. Determine which resources to secure. These can be in the FD, TSD, TDD, JD, PD, or TD.
2. Assign a resource security key (1 through 24) or **public** or **private** to each resource. If you have many resources to be secured, particularly to cross-reference between the CICS resource and the tasks that are using it, consider keeping a written record. You need this when you update the User Definitions. These numbers do not relate in any way to the TSL keys.
3. Change the definitions of all affected resources to reflect their new RSL keys.
4. Change the **RSLCheck** attribute in the Transaction Definitions to **internal**.
5. Change the Terminal Definitions of any attached printers to give them RSL keys if they will be printing out any secure resources.
6. Change User Definitions to include the **RSLKeyList** that is appropriate to each user. Do this by checking which transactions a user can access and which resources that are accessed by the transactions require that the user has an RSL key.
7. Be aware that you can now expect logging of resource security violations, and investigate these when they occur.
8. Let your users know of the new resource security policy.
9. Ensure that CEMT, CECI, and CEDF use resource security checking.

Using an External Security Manager

You can use an External Security Manager (ESM) instead of, or in conjunction with, CICS internal security to control access to CICS resources and transactions. The ESM is a user-supplied program that allows you to define your system's own security mechanism for preventing unauthorized user access to resources from application programs, and the unauthorized initiation of CICS transactions.

How to use an ESM with CICS

By default CICS uses internal security checking. To use an External Security Manager instead of CICS, you must do the following:

- Install the External Security Manager.
- Change the Region Definitions (RD) **ESMLoad** attribute to **yes**.
- Use the RD **ESMModule** attribute to specify the ESM program path and name.

To enable external resource checking for a resource type, set any of the following RD attributes to **external**:

- **FileRSLCheck**
- **JournalRSLCheck**
- **ProgramRSLCheck**
- **TemporaryStorageRSLCheck**
- **TransactionRSLCheck**
- **TransientDataRSLCheck**

Transactions accessing these resources must specify that they are to use the ESM.

The number of security checks for the region that are performed by the ESM depends on the amount of checking that CICS internal security performs. You can use an ESM to either enhance or replace internal security checking.

Information about the CICS resource or transaction is passed to the ESM by CICS. The ESM must use this, and possibly other information, to determine whether authorization is allowed. The CICS system must then be informed, through the interface, of the ESM decision. You must identify the ESM to the CICS system at startup; you possibly also need to identify the CICS system to the ESM.

If you change the ESM during the life of a region, you must perform a warm start to make the ESM valid for all application servers. Otherwise, the changed ESM applies only to tasks that are running under application servers that were started after the ESM was changed. This condition can cause inconsistencies in security within the region.

It is best to design your ESM so that it stores its data separately from the program itself. You can access this data at all times, without needing to restart CICS.

Defining the interface between CICS and an ESM

You can install an ESM module for CICS to invoke automatically when it needs to perform a security check that you have defined as external. CICS supplies an example implementation of this module to demonstrate the interface that you need when writing your own ESM. For information about this supplied module, see the *TXSeries for Multiplatforms Administration Reference*.

Note: CICS passes the user identifiers to the ESM for security checking. In the case where no UD mapping can be found, no principal is passed.

The ESM is a separately compiled module that implements three functions in its interface to the CICS security module: the **Initialization**, the **RSLCheck**, and the **TSLCheck** functions. If this module is required, the **load** subroutine loads it into the CICS system, at CICS startup. The initialization function for an ESM is called **CICS_ESM_Init**.

The use of the three function names is only for reference purposes. The CICS system accesses these functions from the security module indirectly only by address, instead of from a call that names the function. Therefore, you can define the actual names that are used in the ESM module when you implement an alternative ESM. These functions are described in the following list:

- **Initialization** function: The **Initialization** function is called by the CICS startup process to enable the ESM to perform any system-specific initialization and to establish the addresses of the ESM callback functions that perform transaction-level and resource-level security checking. This function is the entry point for a loaded module. The **load** subroutine call returns the address of this function to CICS, so it does not need to be linked into CICS. CICS also supplies this function with data that describes the CICS region. The addresses of the two callback functions that are to perform transaction level and resource level security checking are returned as data.

The following parameters are passed to the **Initialization** function:

- **RegionData**
- **FnBuf**

- **TSLCheck** function: The transaction security callback function is called by CICS to perform external Transaction Security Level (TSL) validation. CICS supplies the function with relevant data that describes the CICS region, user, transaction, terminal, and program.

The following parameters are passed to the **TSLCheck** function:

- **RegionData**
- **ContextData**

- **RSLCheck** function: The resource security callback function is called by CICS to perform external Resource Security Level (RSL) validation. CICS supplies the function with relevant data that describes the CICS region, user, transaction, terminal, program, and resource.

The following parameters are passed to the **RSLCheck** function:

- **RegionData**
- **ContextData**
- **ResourceData**
- **ResourceClass**
- **TaskData**
- **EIB_Fn**

Note: When CEDF is used on a transaction that is defined to use an ESM (**RSLCheck** and **TSLCheck** functions are set to **external**), context and resource data are passed to the ESM for the transaction being that is being diagnosed and for CEDF.

Resource data that is relevant to the ESM call is supplied by CICS as a set of structure reference arguments, one for each relevant resource. All the structures contain at least one common identifier field, the CICS resource name. Each type of structure is made up of various fields that contain information that is appropriate to the type of resource in question. The use of these resource structures allows you to make future changes without breaking the existing ESM code.

ESM function parameters

The ESM functions have the following parameters:

- **RegionData:** This describes the CICS region to the ESM. It contains the system identifier of the CICS region together with the CICS release version number.
- **FbBuf:** This structure is the ESM function buffer and is used to return the addresses of the callback functions that perform ESM validation. This structure enables you to add new ESM functions.
- **ContextData:** This structure contains substructures that describe the current transaction, user, terminal, program, and link region. The program substructure is null for a TSL check because no program is running when this is made. The link region substructure is null unless the transaction was scheduled from a remote system.
- **ResourceData:** CICS passes a variety of different resource definition data to the ESM to enable it to validate access to the particular resource. This data varies with the resource class. A generic resource data value is passed in the ESM call. The ESM uses the **ResourceClass** argument to resolve it to a particular type. The resource classes follow:

- **Transactions**

The following fields are included in the **ResourceData** for transactions:

- Transaction identifier
- Internal/remote identifier
- Remote transaction identifier

- Remote system identifier
- **Programs**
The only field that is included in the **ResourceData** for programs is the Program identifier.
- **Files**
The following fields are included in the **ResourceData** for files:
 - File identifier
 - Internal/remote indicator
 - Remote file identifier
 - Remote system identifier
 - SFS file volume
 - SFS file name
 - SFS file index

You must define the lengths of the SFS filename components that are used in the file data structure that is passed to the ESM here. The volume name is currently defined as 256 characters long, the filename as 256 characters, and the index name as 16 characters.
- **Transient data queues**
The following fields are included in the **ResourceData** for transient data queues:
 - Transient data queue identifier.
 - Transient data queue type. This indicates to the ESM the type of queue and therefore which of the fields (remote transient data queue identifier, remote system identifier, extrapartition name, and indirect name) are valid. It holds one of the values INTRA, EXTRA, REMOTE, or INDIRECT.
 - Remote transient data queue identifier.
 - Remote system identifier.
 - Extrapartition name.
 - Indirect name.
- **Temporary storage queues**
The following fields are included in the **ResourceData** for temporary storage queues:
 - Temporary storage queue identifier.
 - Temporary storage queue type. This indicates to the ESM the type of queue and whether the remote temporary storage queue identifier, and remote system identifier are valid. It holds one of the values MAIN, REMOTE, AUXILIARY, or TEMPLATE.
 - Remote temporary storage queue identifier.
 - Remote system identifier.
- **Journals**
The only field that is included in the **ResourceData** for journals is the Journal identifier.
- **ResourceClass**
This attribute is passed by CICS to the ESM RSL validation function to resolve the type of resource identifier that is to be validated, and the resource-specific data.
- **TaskData**
CICS supplies the ESM with the addresses of two user data areas: the Common Work Area (CWA) and the User Area (TCTUA). These addresses are

passed as pointers in a structure. In conditions in which the task context is not available, it can be supplied as a null pointer.

– **EIB_Fn**

This is a two-byte code for the EXEC CICS operation that is performing a resource access that is obtained from the FN field of the EXEC Interface block (EIB).

CICS supplies the lengths of various identifier and name fields (not including the null byte at the end) to the ESM, at this point.

The length fields and their defaults are shown in Table 27:

Table 27. Default ESM parameter field lengths

Field name	Default length
File identifier length	8 bytes
System identifier length	4 bytes
Transient data queue identifier length	4 bytes
Extrapartition name length	127 bytes
Indirect name length	4 bytes
Temporary storage queue identifier length	8 bytes
Journal identifier length	2 bytes
Program identifier length	8 bytes
Transaction identifier length	4 bytes
Terminal identifier length	4 bytes
User identifier length	8 bytes
Release identifier length	4 bytes

The initialization function returns an integer 0 (zero) to indicate success, and any other value to indicate that an error has occurred.

The indicator that is returned from the **TSLCheck** and **RSLCheck** functions can be one of three possible values:

- **CICS_ESM_RETURN_OK** : This indicates that access to the specified transaction or resource is valid.
- **CICS_ESM_RETURN_INVALID** : This indicates that access to the specified transaction or resource is invalid (that is, a NOTAUTH).
- **CICS_ESM_RETURN_FAIL** : This indicates that the given transaction or resource cannot be validated externally. The message ERZ045014E is issued for the **TSLCheck** function, and ERZ045013E is issued for the **RSLCheck** function. CICS terminates the task and the application server. This means that, possibly, an ERROR has occurred during validation. When the application server terminates, several ERZ087001E messages are written to the **console.nnnnnnn** file. These messages indicate RPC exceptions that are associated with the termination of the application server.

If the task that requested the security check by the ESM holds any crucial locks when the abnormal termination of the application server is signaled, the termination of the task can cause the termination of the region.

Defining an ESM at startup

During initialization of each application server, CICS checks the value of the **ESMLoad** attribute in the Region Definitions (RD). If **ESMLoad** is set to **yes**, CICS loads the ESM module into the application server storage area. The region fails to start if the ESM fails to load. If the return code of the ESM program indicates a failure, CICS terminates the application server and the transaction.

The attribute **ESMModule**, which is defined in the RD, is used as an argument to the load call. **ESMModule** is either an absolute or a relative pathname to the ESM module. If a relative pathname is used, the library path environment variable must contain the directory that contains the ESM module. Table 28 lists the library path environment variables for each platform:

Table 28. Library path environment variables

Platform	Environment variable
AIX	LIBPATH
HP-UX	SHLIB_PATH
Solaris	LD_LIBRARY_PATH

The load call returns the entry point address of the ESM initialization function. You must define this entry point address when you are building the ESM module. The CICS security module calls the initialization function of the ESM, which allows the ESM to perform any specific initialization. The ESM initialization function uses a return code to indicate the success of the initialization. If the installation of the ESM fails, the startup process is not completed, and the region shuts down.

If **ESMLoad** is set to **no**, any transactions or resources that have been defined as requiring external security default to using internal security checking and a warning message is issued. Therefore, it is recommended that the **TSLKey** and **RSLKey** attributes for any transactions or resources that can be affected by this are set to values that are enough to provide protection in case the region is ever started without the ESM.

Defining CICS to an ESM

You might need to define the components of a CICS system to an ESM. The definition method is the responsibility of the ESM, but you might be required to ensure that this definition has been completed before the ESM can be used for CICS security checking.

Using the ESM with CICS system transactions

CICS runs some of its system transactions under a CICS user ID that matches the region name, and runs other system transactions under a null CICS user ID. Ensure that your ESM does not prohibit access to these transactions.

Part 2. Setting up resources for use within a CICS region

This part of the guide describes how to use resource definitions to increase the functions of your CICS system. It includes methods that enable your CICS system to interact with application data files through File Definitions (FD), Product Definitions (PD), and Transaction Definitions (TD). It also describes how to access user exit programs and third-party products, such as relational databases, Micro Focus Server Express COBOL, and IBM WebSphere MQ.

The topics are covered in the following sequence:

- Chapter 8, "Working with CICS resource definitions," on page 115
- Chapter 9, "Working with Program Definitions and Transaction Definitions," on page 127
- Chapter 10, "Working with application data files in CICS," on page 133
- Chapter 11, "Configuring CICS for use with relational databases," on page 143
- Chapter 12, "Preparing CICS for use with Micro Focus Server Express COBOL," on page 175
- Chapter 13, "Preparing CICS for use with ACUCOBOL-GT," on page 177
- Chapter 14, "Working with WebSphere MQ," on page 179
- Chapter 15, "Customizing CICS with user exits and user programs," on page 185

Chapter 8. Working with CICS resource definitions

A CICS *resource* is any facility or component of a CICS system that is required to perform a task. To be used by CICS, each resource requires a resource definition. “Identifying resource definition classes” on page 5 shows the classification of the definable resources that CICS uses. Many resource definitions are automatically set when region configuration commands are initially issued. This chapter discusses the commands that are used to add, modify, and delete resource definitions from your CICS region.

“Understanding how the resource definition databases are used” explains the use of resource definitions in the permanent, autostart, and runtime databases.

“Using CICS resource management commands to update resource definitions” on page 116 explains how to change resource definitions in the permanent, autostart, and runtime databases.

“Defining resources to be selectively installed” on page 124 explains how to define resources to be selectively installed from a cold start or into a runtime database.

“Configuring for shared resource definitions” on page 125 explains how to share Region Definitions, Communications Definitions, definitions that can be shipped as functions, and the Region Definitions that the SFS server uses.

Understanding how the resource definition databases are used

Three databases are used to store resource definitions: *permanent*, *autostart*, and *runtime*.

The *permanent database* contains the definitions that define the state of a region and its resources when it is cold started. This database is initially created when the region is set up.

When a region is cold started, a new *autostart database* and *runtime database* are concurrently installed with a subset of resource definitions from the permanent database. This subset includes:

- Those definitions with the **ActivateOnStartup** attribute that is in the Region Definitions (RD) stanza set to **yes**.
- Those definitions with the **Group** attribute set to a group name that is listed in the Region Definitions (RD) **Group** attribute. For example, if the **Group** attribute of a transaction is set to **Sales** and the list of groups that are defined with the region’s **Group** attribute contains **Sales**, the definition for that transaction is installed into the autostart and runtime databases when the region is cold started, whether or not the **ActivateOnStartup** attribute is set to **yes** for the transaction.

When a region is warm started, the runtime database is installed from resource definitions that are in the autostart database. The region runs from the runtime database, which exists only when the region is running.

The permanent and autostart databases are in the `/var/cics_regions/regionName/database` directory.

Using CICS resource management commands to update resource definitions

This topic describes how to use the CICS resource management commands to update resource definitions in the permanent and runtime databases. “Updating resource definitions in a running region” tells you which procedures to use for adding, changing, installing, and deleting region runtime resources.

Understanding the scope of commands affecting resource definitions

CICS resource management commands affect definitions for all classes of resources, except Region Definitions (RD) and Monitoring Definitions (MD). (See “Identifying exceptions to when resource definitions can be changed” on page 123 for discussion on the exceptions.)

The resource management commands support the **-c** option, which enables identification of the resource definition class. The following notations can be associated with the **-c** option in a resource management command to identify the resource class:

- **cd**: Communications Definitions (CD)
- **fd**: File Definitions (FD)
- **gd**: Gateway Definitions (GD)
- **jd**: Journal Definitions (JD)
- **ld**: Listener Definitions (LD)
- **pd**: Program Definitions (PD)
- **td**: Transaction Definitions (TD)
- **tdd**: Transient Data Definitions (TDD)
- **tsd**: Temporary Storage Definitions (TSD)
- **ud**: User Definitions (UD)
- **wd**: Terminal Definitions (WD)
- **xad**: Product Definitions (XAD)

To identify the CICS database in which the definition is to be installed, supply the **-P** (permanent), **-R** (runtime), or **-B** (both permanent and runtime) options with the resource management command.

The region that is to be affected by the resource management command is identified by supplying the **-r** (lowercase) option and the name of the affected region.

Unless an attribute value is specifically defined with the resource management command, the default values for the attributes that are associated with each class of resource definitions are assigned. When the resource management command is issued, multiple attributes can be assigned values in a single command. See the *TXSeries for Multiplatforms Administration Reference* for the list of attributes that are associated with each resource definition class, and the values that are valid for the attributes.

Updating resource definitions in a running region

You can use CICS resource management commands to change the resource definitions in a running region in several ways:

- If the resource definition does not exist in the permanent database, you can use the **cicsadd** command with the **-B** option to add it to the permanent database and to the runtime. See “Adding a new resource definition to a running region.”
- To change a resource definition that was installed in the runtime, you must first delete it from the runtime, then make the change in the permanent database and install the definition in back into the runtime. See “Changing a resource definition in the permanent, autostart, and runtime databases.”
- If a resource is defined with the **ActivateOnStartup** attribute set to **no** and does not belong to a group that is defined to the region, it is not installed into the runtime when the region is cold started. To install it into the runtime, see “Installing a resource definition from the permanent database” on page 118.
- To install or reinstall a group of definitions that have the **Group** attribute value set to a group name, or to reinstall all definitions that have the **ActivateOnStartup** attribute set to **yes**, see “Installing a group of definitions into the runtime database” on page 119.
- To delete a resource definition from the runtime only, see “Deleting a resource definition only from the runtime database” on page 118.
- To use CEMT to administer resource definitions in a running region, see “Using CEMT to manage runtime resources” on page 119.
- To use the CICS INQUIRE and SET commands to administer resource definitions, see “Using the EXEC CICS INQUIRE and the EXEC CICS SET commands” on page 123.

Adding a new resource definition to a running region

Use the following procedure to add a new resource definition to the permanent, autostart, and runtime databases:

```
cicsadd -c class -r regionName -B key \
      attribute=value \
      .
      .
      .
      attribute=value
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

If, for some reason, the runtime database is not updated (for example, if the region is not running), the autostart database is not updated either. However, the permanent database is updated.

If this happens, fix the problem (for example, start the region), then follow the directions that are given in “Installing a resource definition from the permanent database” on page 118 instead.

Changing a resource definition in the permanent, autostart, and runtime databases

Use this procedure to change a resource definition in the permanent, autostart, and runtime databases.

1. Delete the resource definition entry from the runtime database. Enter the command:

```
cicsdelete -c class -r regionName -R key
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

2. Change the resource definition entry in the permanent database and install it in the runtime. Enter the command:

```
cicsupdate -c class -r regionName -B key \  
attribute=value \  
.  
.  
.  
attribute=value
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

Note: When a resource definition already exists in the runtime database, you must delete that existing runtime definition, before updating the definition that is in the permanent database.

Installing a resource definition from the permanent database

If a resource is defined with the **ActivateOnStartup** attribute set to **no** and does not belong to a group defined to the region, it is not installed from the permanent database into the runtime database when the region is cold started.

Use these procedures to install a resource definition from the permanent database into the runtime database when the definition was not installed into the runtime at the cold start of the region.

- To add a resource definition entry that is in the permanent database (but has not been installed) to the runtime database without changing the existing definition, enter the command:

```
cicsupdate -c class -r regionName -B key
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

- To change the default attribute values for the resource definition in the permanent database while adding it to the autostart and runtime databases, enter the command:

```
cicsadd -c class -r regionName -B key \  
attribute=value \  
.  
.  
.  
attribute=value
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

If for some reason the runtime database is not updated (for example, if the region is not running), the autostart database is not updated either.

If this happens, fix the problem (for example, start the region) then repeat the procedure.

Deleting a resource definition only from the runtime database

Use this procedure to delete a resource definition from only the runtime database. You cannot use this procedure for Region Definitions (RD).

```
cicsdelete -c class -r regionName -R key
```

where *key* is the key name for the resource definition entry. See “Understanding the scope of commands affecting resource definitions” on page 116 for an explanation of the other options that are associated with this command.

Installing a group of definitions into the runtime database

Use these procedures to install a group of resource definitions from the permanent database into the runtime database. To set up groups to be installed, see “Defining resources to be selectively installed” on page 124.

- To install resource definitions that have been defined with the **Group** attribute set to a group name, enter the command:

```
cicsinstall -r regionName -g groupName [groupName...]
```

- To reinstall resource definitions that have the **ActivateOnStartup** attribute set to **yes**, enter the command:

```
cicsinstall -r regionName -a
```

Note: This procedure overrides any changes that were made to these resource definitions with CEMT or EXEC CICS SET.

Using CEMT to manage runtime resources

This section provides an overview of how to use the CICS-supplied CEMT transaction to view and modify the runtime resources in a CICS region.

You can use the CEMT transaction to perform the following tasks:

- Control the journaling, monitoring, trace, and dump facilities
- Query the values of attributes that belong to tasks, terminals, programs, and other CICS resources, by using CEMT INQUIRE
- Reset these values where necessary, by using CEMT SET

Changes that are made by using the CEMT transaction affect only the runtime environment and are lost whenever the region is shut down. To make permanent changes to resource definitions, use the CICS resource management commands.

The CEMT transaction affects only specific resources and a specific subset of the attributes that are applied to them. The details are described in the *TXSeries for Multiplatforms Administration Reference*.

You can use the CEMT transaction to inquire about or set attributes for the following resources and facilities:

- System parameters (subset of Region Definitions (RD))
- Transactions
- Programs
- Maps
- Terminals
- Terminals using autoinstall
- Files
- Transient data queues
- Tasks and task classes
- Trace facilities
- Journal facilities
- Monitoring facilities
- Dump facilities

Using CEMT

Note: Before you can use the CEMT transaction for resources, the value that is in the **TSLKeyList** attribute in your User Definition (UD) entry must match the transaction security key that is defined for CEMT in the Transaction Definitions. Also, the value that is in the **RSLKeyList** attribute in your UD must match the resource security key that is defined for the resource that you need to access.

Invoke CEMT from the CICS command line. For details of how to start a local terminal session, see *TXSeries for Multiplatforms Installation Guide*.

You can enter a CEMT request in stages or, if you know all the parameters, as a complete request. The CEMT transaction checks the syntax of requests and diagnoses errors. If your request is syntactically correct, CICS performs it immediately. If syntax errors exist, CICS displays the correct syntax for your request.

After you have the CEMT transaction running, you do not have to include CEMT at the beginning of subsequent requests because it is implied until you exit from CEMT. To exit from CEMT, use **PF3**.

Starting CEMT

To start the CEMT transaction, type CEMT on the command line of your screen, and press **Enter**. CICS displays the following:

```
STATUS: ENTER ONE OF THE FOLLOWING
Discard
Inquire
Perform
Set
```

Enter one of the three options:

- If you want to inquire about a resource, enter INQUIRE (or any abbreviation) on the command line.
- If you want to perform a function that is not related to resource status, enter PERFORM (or any abbreviation) on the command line. (The only such function available is PERFORM SNAP).
- If you want to set, or change, the value of a resource, enter SET (or any abbreviation) on the command line.

If you enter any of the three keywords without additional parameters, CICS displays the options that are available with the command. Each option is displayed with its minimum permitted abbreviation in uppercase characters and the remainder in lowercase. For example, if you enter INQ, you see the following screen:


```

I
STATUS:  ENTER ONE OF THE FOLLOWING OR HIT ENTER FOR SYSTEM PARAMETERS

AUTInstmode      TClass
AUXtrace         TRACe
DUMP             TRAnsaction
DUMPOptions
FEConnection
FENode
FEPOol
FEPropset
FETarget
File
TDqueue
TErминаl
Netname
Program
TAsk
APPLID=region7
PF 1 HELP        3 END                                9 MSG

```

Figure 9. CEMT INQ

You can inquire about any of the displayed resources by typing its keyword after INQ on the command line. For example, entering CEMT INQ PROGRAM displays the name of each program, what language it is written in, and whether it is enabled or disabled.

Note: If you press **Enter** without specifying an option, a complete list of the system parameters is displayed. You can find an explanation of these in the *TXSeries for Multiplatforms Administration Reference*.

The CEMT screen layout

The CEMT screen contains three distinct areas. These are:

- Command input (the first line of the screen)
- Information (19 lines on a 24 x 80 screen)
- PF key values (the last line of the screen)

Command input: Use this line to enter any of the CEMT requests described in this section, for example:

```
INQ FILE(*)
```

Information: This is the main area of the screen between the command input area and the PF key values. CICS uses this area to display the options that you can use in the current command context, or information that you have requested about resources.

Program function (PF) key values: The following list explains the meanings of the program function (PF) keys, which appear on the bottom line of the CEMT screen. If your keyboard does not have PF keys, you can use the **Tab** key to position the cursor under the required instruction on the screen keys, and press **Enter**.

PF1 Displays help.

PF3 Exits from CEMT, so that CICS terminates your CEMT transaction and ends the session. However, if you use this key when modifying your screen, for example, by keying in a new command or by overtyping an old command, CICS ignores the request and does not end your session.

- PF7** Scrolls back half a page.
- PF8** Scrolls forward half a page.
- PF9** Expands messages. If CICS generates more than one message in response to your request, CICS displays a message near the bottom of the screen telling you how many messages have been generated. **PF9** allows you to display all the messages on a separate screen. To go back to your original screen, press **Enter**.
- PF10** Scrolls back a whole page.
- PF11** Scrolls forward a whole page.

Scrolling symbol (+ sign): A plus sign (+) on the first or last line of a screen tells you that more data precedes or follows the current screen. Scrolling backward displays preceding data, and scrolling forward displays following data.

Entering CEMT requests

CEMT requests can be entered in several ways, as follows:

The ALL parameter: Many of the request formats contain the ALL parameter. For INQUIRE, ALL is the default value. For SET, it is not. For example:

INQ TERMINAL

Shows the attributes of all terminals (default is ALL).

INQ TERMINAL ALL

Also shows the attributes of all terminals.

SET TERMINAL PRI(4)

An invalid command (default is not ALL).

SET TERMINAL ALL PRI(4)

Sets the priority of all terminals to 4.

Specifying a family of resources (using asterisk and plus symbols): You can use the symbols asterisk (*) and plus (+) as part of an identifier to specify a family of resources. The asterisk (*) represents any number of characters (including none), and the plus sign (+) represents a single character. For example:

- A* specifies all identifiers of any length beginning with A.
- TERM00+ specifies all 7-character identifiers beginning with TERM00.

Subdefining resource groups: When using INQUIRE, you can request a subgroup of resources by specifying one or more attributes of the resource. For example, the following command displays the status of all terminals that are in service:

```
INQ TERMINAL ALL INSERVICE
```

You cannot define a subgroup in this way when using SET. For example, the following command puts *all* terminals in service and also sets their priority to 4, instead of setting the priority of all those that are currently in service.

```
SET TERMINAL ALL INSERVICE PRIORITY(4)
```

However, you can set values for subgroups of resources by using INQ and overtyping the values that are displayed on the screen.

Overtyping: You can move the cursor to any alterable field on a screen and change its contents by overtyping. When you next press **Enter**, CICS reads the

contents of all the fields that you have changed and performs any valid operations that the changes require. If you make an invalid change, CICS displays an error message and does not change the field.

The values for each field are not case sensitive, so you can overwrite fields with lowercase or uppercase characters.

The Tab key: The fields that you can change are different in each screen, but you can detect them by pressing the **Tab** key repeatedly. This moves the cursor from one field to the next, and you can then overwrite the field with the appropriate information.

Although you can move to the PF key fields at the bottom of the screen by pressing the **Tab** key, overtyping these fields has no effect. If you press **Enter** when you are in one of these fields, the result is the action of that PF key.

Some screens contain blank fields that allow you to specify parameters, such as NEWCOPY and PURGE, that are not part of the status of a resource. A status that begins with “NO”, such as NOREAD, is also represented by a blank field. You can use the Tab key to detect these fields.

The information in subsequent sections tells you which attributes can be set for each resource and facility.

Using the EXEC CICS INQUIRE and the EXEC CICS SET commands

CICS provides the EXEC CICS INQUIRE and EXEC CICS SET commands, which allow you to examine and change the CICS region’s resource definitions from within a user-written system program that is defined to CICS. These commands perform functions that are very similar to those that the CEMT transaction performs.

CICS provides this facility primarily for user-written programs (for example to customize the CEMT transaction), and not for general applications.

Changes that are made by using the EXEC CICS SET command apply only for the duration of the current CICS session because they apply only to the runtime database for the region.

You can find detailed information about using the commands in the *TXSeries for Multiplatforms Administration Reference*.

Identifying exceptions to when resource definitions can be changed

The exceptions to when you can change definitions in the databases, are:

- The Region Definitions (RD) can be changed only in the permanent database, except for the following attributes:
 - **MaxServer**, **MinServer**, **SysDump**, **PCDump**, and **ABDump** can be changed only in the runtime database by using the CEMT facility.
 - **StatsRecord** can be changed only in the runtime database by using the EXEC CICS SET STATISTICS RECORDING command.

- The Monitoring Definitions (MD) can be changed only in the permanent database, except for the **MonitorStatus** attribute. You can change the **MonitorStatus** attribute only in the runtime database by using the CEMT facility.
- The Product Definitions (XAD) are loaded from the permanent database on a warm start. This is unlike all other resource classes. The XAD are loaded from the autostart database only when an emergency restart is performed.
- Some changes can be made only to the runtime database with the CEMT facility and with the EXEC CICS SET command. However, any changes made to the runtime database are lost when the region is shut down.

For more information about the commands used to change resource definitions in the permanent, autostart, and runtime databases, see the *TXSeries for Multiplatforms Administration Reference*.

Defining resources to be selectively installed

You can define resources to be selectively installed from the permanent database into the runtime. Use the following methods:

- Define a group of resources.
To define a group of resources, set the **Group** attribute to the name of the group. You can do this for the following resource classes:
 - Communications Definitions (CD)
 - File Definitions (FD)
 - Gateway Definitions (GD)
 - Journal Definitions (JD)
 - Listener Definitions (LD)
 - Product Definitions (XAD)
 - Program Definitions (PD)
 - Terminal Definitions (WD)
 - Temporary Storage Definitions (TSD)
 - Transaction Definitions (TD)
 - Transient Data Definitions (TDD)

The group can be installed either at a cold start of the region or at runtime.
 - Define groups that are to be installed on a cold start.
To install the group into the runtime when the region is cold started, define the group name to the region with the Region Definitions (RD) **Groups** attribute. This attribute can contain a list, for example, **Sales, Payroll**, and so on.
 - Define groups that are to be installed in the runtime.
If you do not want the group installed when the region is cold started, but you want to be able to install it into a running region, do not include the group name in the RD **Groups** attribute. You can then use the **cicsinstall** command as described in “Installing a group of definitions into the runtime database” on page 119.
- Note:** This command can also be used to reinstall a group of resources or to reinstall all resource definitions with the **ActivateOnStartup** attribute set to **yes**.
- Use the **ActivateOnStartup** attribute.

Set the **ActivateOnStartup** attribute to **no** and the **Group** attribute to either blank or to a group that is not defined to the region. These settings prevent a resource definition entry from being installed into the runtime when the region is cold started. To install the resource definition into a running region, see “Installing a resource definition from the permanent database” on page 118.

Configuring for shared resource definitions

Regions can share the whole resource definitions database or definitions for one or more resource class. Sharing the whole database is convenient for setting up a test region, but if some resource classes vary greatly, it can be impractical to share the whole database between production regions. It is more convenient to set up and share a group containing only the most common resource classes.

A shared resource database can be set up in two ways:

- Delete a region’s whole database directory or individual resource class files, then use the **ln** command to create a link to another region’s database directory or files.
- When creating regions, use the **cicsexport** and **cicsimport** commands to create *symbolic links* for specified database classes automatically. To do this, when creating the archive copy (by using the **cicsexport** command), specify the classes for which you want to create links. If you then use the **cicsimport** command to use the definitions in that archive for a new region, the region will already have links from it to the specified database class subdirectories. For more information about these two commands, see “Defining a region using the **cicsexport** and **cicsimport** commands” on page 273 and the *TXSeries for Multiplatforms Administration Reference*.

Sharing Region Definitions (RD)

Each region must have a unique **LocalSysId**, which is an attribute in the Region Definitions (RD). However, if regions are sharing RDs, each region is automatically given the same **LocalSysId**. To avoid this, ensure that the **LocalSysId** attribute is overridden when the region is started. See “Starting a region from the command line” on page 280 for more information.

All regions that share a set of Region Definitions must have been created with the same **-g groupId** parameter specified (with the **cicsimport** and **cicsdefault** commands) or with none at all. Regions that are associated with different group names cannot share resources; they do not have access to the databases and therefore will not start.

Sharing SFS Region Definitions (RD)

Use the shorthand notation **%R** when specifying filenames. The notation **%R** expands to the local region name. If you do not use **%R**, all your SFS files will be shared. See “Using shorthand in the resource definitions” on page 8 for more information about how to use shorthand notation.

Sharing Communications Definitions (CD)

When the region is started, a warning might be displayed that a definition exists for the local system. You can ignore this warning.

You must create a set of CDs for each region that shares the same database.

Sharing definitions that can be function shipped

When you specify the **RemoteName** attribute for resource definitions that can be shipped as functions (TD, TDQ, TSQ, FD, PD), use a blank or use the name of the resource as it is known on the machine on which it resides.

RemoteName is ignored on the local system. Complete the attribute list for both remote and local information because each is used differently.

Chapter 9. Working with Program Definitions and Transaction Definitions

“Configuring Program Definitions (PD)” describes how to enable Program Definitions for application programs.

“Autoinstalling Program Definitions (PD)” on page 129 describes how to autoinstall Program Definitions.

“Configuring Transaction Definitions (TD)” on page 130 describes how to enable Transaction Definitions for use by your CICS system.

The chapter assumes that you have followed the instructions that are given in the *TXSeries for Multiplatforms Installation Guide* to set up a default CICS system.

Configuring Program Definitions (PD)

Several types of programs are used in CICS. The *TXSeries for Multiplatforms Application Programming Guide* describes application programs that use the CICS API and programs that are used for presentation services. Chapter 15, “Customizing CICS with user exits and user programs,” on page 185 describes user-written programs that are invoked from within CICS. These programs all require Program Definitions (PD).

PD define all the programs that CICS can process in the region. Each entry within the PD contains control information that CICS uses to identify and run a program.

For more information about PD and their attributes, see the *TXSeries for Multiplatforms Administration Reference*.

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Configuring the Program Definitions (PD)

The following command examples show how to work with the Program Definitions (PD) in the permanent and runtime databases.

- Adding a new PD:

This example shows how to add a PD entry for a program named **sales1**. The program is in **/var/sales/bin**. The defaults are assumed for all other PD attributes.

```
cicsadd -c pd -r regionA -P sales1 PathName='/var/sales/bin/sales1'
```

If the region is running, and you want to add the definition to the runtime in addition to the permanent database, use the **-B** (both permanent and runtime) option, as follows:

```
cicsadd -c pd -r regionA -B sales1 PathName='/var/sales/bin/sales1'
```

Note: Alternatively, you can autoinstall a program definition by using user exit 13. See “Autoinstalling Program Definitions (PD)” on page 129 for more

details. However, an autoinstalled program definition is added only to the runtime database. Therefore, that definition exists only until the corresponding region is active.

- Changing an existing PD:

This example shows how to change a PD entry status in the permanent database.

```
cicsupdate -c pd -r regionA -P sales1 EnableStatus=disabled
```

- Deleting a PD:

This example shows how to delete a PD entry from the permanent database.

```
cicsdelete -c pd -r regionA -P sales1
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- Viewing a PD entry:

This example shows how to view a PD entry:

```
cicsget -c pd -r regionA sales1
```

Working with PD in the runtime database

To work with the Program Definitions (PD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change Program Definitions (PD) while the region is running. Any changes that are made by using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions at the next restart.

CEMT lets you:

- Inquire about runtime Program Definitions (PD) or runtime programs with a specific source language
- Enable or disable a runtime program or a mapset
- Force CICS to use the latest version of a program or mapset

For more information, see the *TXSeries for Multiplatforms Administration Reference*.

Program Definitions and setting up Basic Mapping Support for different languages

To have the same map and mapset available for use in different languages, you can take the map source and translate it to the new language. You then process the map with the **cicsmap** command to ensure that the physical map is placed in the relevant locale directory under `/var/cics_regions/regionName/maps`.

For example, if your original mapset is in English, (that is, the **en_US** language file on the AIX platform, the **En_US** language file on the HP-UX platform, or **C** on the Solaris platform), and you want to have the same mapset available in French, follow these steps:

1. Use the **cicsmap** command to place the physical mapset into the language directory, (that is, the **en_US** directory on the AIX platform, the **En_US** directory on the HP-UX platform, or the **C** directory on the Solaris platform). Refer to the description of the **-f locale** option that is given in the *TXSeries for Multiplatforms Application Programming Guide*.
2. For a French version of the map, use **cicsmap** again, and verify that the correct locale is used during processing. Ensure that the physical map is placed into the directory for that locale. Create a new directory if one does not already exist.

3. In the Program Definitions (PD) for the map, you must set the **PathName** attribute for the mapset to the mapset name only, so that CICS searches the relevant directories instead of specifying an explicit directory path.

After you have done this, based upon the LANG environment variable of the **cicslterm** client or Telnet server, the appropriate map is displayed.

Note that although the physical maps are possibly different, the logical maps remain the same.

See “Setting the language for the region” on page 21 for information about how to set up languages for the region. See “Setting the language for the user’s client connection” on page 15 for information about how to set up language for the client. See Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11 for a list of the languages that are available for use with CICS.

Autoinstalling Program Definitions (PD)

Note: In this section, the term “program” refers to programs and to mapsets, unless otherwise stated.

CICS program definitions can be autoinstalled. A program is said to be autoinstalled, when it has no permanent definition and has been dynamically defined by CICS in the runtime database.

A program is autoinstalled when:

- These commands reference a previously undefined program:
EXEC CICS LINK
EXEC CICS LOAD
EXEC CICS XCTL
- A previously undefined program is run in a transaction.
- EXEC CICS SEND MAP and EXEC CICS RECEIVE MAP refer to a previously undefined mapset.

Program autoinstall is supported in CICS through user exit 13. With user exit 13, a user can implement the program autoinstall control program, which sets values for the fields that are in the program definition stanza. (See the *CICS Administration Reference* for more details about user exits). If the user does not specify a value for any field, the autoinstall control program defaults to the value that is in the default (or NULL) entry in the program definition stanza. The user can change the value for all fields except the Program name (the name of the program definition entry) and the ProgType. These fields are set internally to CICS. Except for the following four attributes, the user has access to all other fields in the program definition.

AmendCounter

For CICS internal use only.

ResourceDescription

The description of the resource. More relevant for a permanent entry.

ActivateOnStartUp

Specifies whether resource definition is to be copied from the permanent database to the runtime database each time the region is cold started.
Suitable for a permanent entry.

Permanent

Specifies whether amending or deleting a permanent database entry is permitted. Suitable for a permanent entry.

Those four attributes are initialized to the default values when the program definition is autoinstalled.

An autoinstalled program is added only to the CICS runtime program definition database. It therefore no longer exists when the CICS region is recycled.

Note: When the program autoinstall feature is used to install a map, CICS first makes a terminal-specific mapset name by adding to the plain mapset name a suffix that represents the terminal that is to be used. CICS then searches for the mapset. If it cannot find the terminal-specific mapset on the filesystem, it tries to autoinstall the plain mapset name. The user exit can be called in both these conditions, but such an action can cause unnecessary multiple definitions of the same mapset.

See the *CICS Administration Reference* for more details about the program autoinstall facility.

Autoinstall of a program definition might fail if:

- The default program definition entry is missing from the PD stanza
- CICS cannot create a runtime database entry for this program

These operands relate to program autoinstall on the EXEC CICS SET SYSTEM and EXEC CICS INQUIRE SYSTEM commands:

PROGAUTOEXIT

Specifies the name of the program autoinstall exit.

PROGAUTOINST

Specifies whether the program autoinstall function is active or inactive.

For more information about the EXEC CICS commands, see the *CICS Administration Reference*.

Configuring Transaction Definitions (TD)

A transaction in CICS is a unit of processing that invokes CICS applications. A transaction is initiated by a single request, often from a terminal. The request usually uses a four-character transaction identifier.

Transaction Definitions (TD) define all the transactions that CICS can process in the region. Each entry within the TD contains control information that CICS uses to identify and run a transaction.

For more information about Transaction Definitions (TD) and their attributes, see the *TXSeries for Multiplatforms Administration Reference*.

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Configuring the Transaction Definitions (TD) in the permanent database

The following command examples show how to add, change, delete, or view the Transaction Definitions (TD) in the permanent database. Full descriptions of Transaction Definitions (TD) and their attributes, including default values, can be found in the *TXSeries for Multiplatforms Administration Reference*.

- Adding a new TD:

This example shows how to add a TD entry with a key of **TRN1** that is in the group **Sales**. The defaults are assumed for all other TD attributes:

```
cicsadd -c td -r regionA -P TRN1 GroupName='Sales'
```

If the region's **Groups** attribute list includes **Sales**, this command installs the entry for this transaction into the runtime database at the next cold start (see "Defining resources to be selectively installed" on page 124). If the region's **Groups** attribute list does not include **Sales**, you must use the **cicsinstall** command to install the TD entry into the runtime (see "Installing a group of definitions into the runtime database" on page 119).

If you want to add the definition to the runtime and permanent databases simultaneously, ensure that the region is running and use the **-B** (both permanent and runtime) option, as follows:

```
cicsadd -c td -r regionA -B TRN1 GroupName='Sales'
```

- Changing an existing TD:

This example shows how to change a TD entry so that it is not installed into the runtime the next time the region is cold started:

```
cicsupdate -c td -r regionA -P TRN1 ActivateOnStartup=no GroupName=''
```

To install the definition into the runtime, see "Installing a resource definition from the permanent database" on page 118.

Note: If the **GroupName** attribute contains a group name that is included in the region's **Groups** list, the TD entry is installed regardless of the value of the **ActivateOnStartup** attribute.

- Deleting a TD entry:

This example shows how to delete a TD entry:

```
cicsdelete -c td -r regionA -P TRN1
```

To delete the entry from the runtime, see "Deleting a resource definition only from the runtime database" on page 118.

- Viewing a TD entry:

This example shows how to view a TD entry:

```
cicsget -c td -r regionA TRN1
```

Working with TD in the runtime database

To work with TD in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change TD while the region is running. Any changes that are made by using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions at the next restart. Use CEMT to:

- View the runtime TD
- Set the priority of a runtime transaction
- Enable or disable a runtime transaction

For more information, see the *TXSeries for Multiplatforms Administration Reference*. To write CICS application programs to manage the status or operation of

transactions in the runtime, see the information about the INQUIRE and SET commands that is given in the *TXSeries for Multiplatforms Application Programming Reference*.

Configuring for the in-doubt condition

Intercommunication often makes use of a *two-phase commit* protocol. This protocol is used to coordinate changes to recoverable resources. In the first phase, all resource managers are asked to prepare their work; in the second phase they are requested to either commit or back out (roll back) changes to data.

Connection or system errors can occur while CICS is processing a two-phase commit for a transaction. Some types of errors cause the transaction to wait until either you correct the errors or they are corrected by CICS itself. These waiting transactions are in an *in-doubt* condition.

If you do not want the transaction to wait for the error to be corrected, do the following:

1. Use the Transaction Definitions (TD) **InDoubt** attribute to specify how the condition is to be handled. When a transaction is waiting between the phases of the two-phase commit process and a CEMT SET TASK FORCEPURGE is issued against the transaction, CICS uses this attribute to determine whether to commit or back out any changes that the transaction made before the wait occurred. Two values are possible for the **InDoubt** attribute:
 - **wait_commit** (commit changes)
 - **wait_backout** (back out changes)
2. Use CEMT INQUIRE TASK INDOUBT to determine whether the transaction is in the in-doubt condition.
3. Use CEMT SET TASK FORCEPURGE to resolve the in-doubt condition according to the TD **InDoubt** attribute setting.

Chapter 10. Working with application data files in CICS

The CICS file management system consists of a basic file structure, queues, and the user's data files. This complete system is created in stages as the CICS system is configured. Chapter 4, "Working with a file manager in a CICS region," on page 23 discusses the creation of the basic file structure for use by CICS to manage files and queues. Chapter 5, "Working with CICS queues," on page 59 discusses the addition of queues to the file management structure. This chapter addresses the use of data files created to hold data resulting from a user's applications.

"Configuring the File Definitions (FD)" describes the procedures for working with the user's data files and shows how to add the user's data files to the file manager by using either a Schema File Definitions (SCD) stanza or a diagnostic tool that is available through CICS.

The FD (one for each data file) describes how the file is to be used by the application program. Although the FD is not used in the configuration process, it must exist in order for CICS to use the file. See "Configuring the File Definitions (FD) in the permanent and runtime databases" on page 135 for more information.

Configuring the File Definitions (FD)

CICS provides an interface to a generic record-oriented file manager. If you are moving to IBM CICS on the AIX, HP-UX, or Solaris platform from an IBM mainframe-based CICS platform, you are probably familiar with *Virtual Storage Access Method (VSAM)* data sets. VSAM is an access method for direct or sequential processing of fixed-length and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequenced), in the physical sequence in which they are written on the data set or file (entry sequenced), or by relative-record number.

VSAM is not supported by your operating system. On Open Systems, CICS emulates VSAM by using SFS or DB2 files.

Procedures used when working with data files

"Creating File Definitions (FD) for record types" on page 134, "Configuring the File Definitions (FD) in the permanent and runtime databases" on page 135, and "Working with FD in the runtime database" on page 136 describe how to create, associate, change, delete, or view the FD for a file in the permanent and runtime databases.

Attention: Before changing an existing FD, ensure that the file itself is closed.

You also need to add the files to the file manager. You can add files to a file manager by using either of two methods:

- Create a file schematic and import files into it.
 - To set up a schematic of a file that can help you *import* files to the file manager, follow the instructions that are given in "Adding files to the file manager by using Schema File Definitions (SCD)" on page 136. If you are using file schematics, and you are going to add the files to the file manager, follow the instructions that are given in "Setting up the data files on DB2

with the `cicsdb2import` command” on page 138 or “Setting up the data files on an SFS server with the `cicssfsimport` command” on page 137.

- Use a diagnostic tool to add files to the file manager.
 - Diagnostic tools are available for DB2 files and for SFS files to add the files to the file manager. “Adding files to the file manager by using the diagnostic tools” on page 139 describes how to add DB2 or SFS files to the file manager. To use these tools to add files to the file manager, follow the instructions that are given in “Setting up the data files on DB2 with the `cicsddt` diagnostic tool” on page 140 or “Setting up the data files on SFS using the `cicssdt` diagnostic tool” on page 139.

If the file is remote, see the *TXSeries for Multiplatforms Intercommunication Guide*.

Creating File Definitions (FD) for record types

See the *TXSeries for Multiplatforms Administration Reference* for complete information about the various File Definitions (FD) and their attributes. The following attributes associate a CICS FD with file management:

- **FileServer:** The name of the SFS server on which this file is located. This attribute is ignored if DB2 is used as the file manager because the region can use only one DB2 database for file management.
- **BaseName:** The name of the file.
- **IndexName:** The name of the index.

You must create a separate FD for each index by which you want to access a file. Here are some examples of various file definitions:

- A key-sequenced data set (KSDS) file:

Assume that you have created a KSDS file named **personnel** whose primary index is **number** and whose secondary index is **name**. Create two FD entries, one allowing access to the file by the primary index, and the other allowing access to the secondary index. The definitions contain:

```
FD key      : PERSNUMB
BaseName    : personnel
IndexName   : number
```

```
FD key      : PERSNAME
BaseName    : personnel
IndexName   : name
```

- An entry-sequenced data set (ESDS) file:

You can create definitions for ESDS files in a similar way to that which you used for KSDS files. Assume that you have created an ESDS file named **history** whose primary index is **sequence** and whose secondary index is **owner**. You can create file definitions that allow access with these two indexes:

```
FD key      : HISTSEQ
BaseName    : history
IndexName   : sequence
```

```
FD key      : HISTOWN
BaseName    : history
IndexName   : owner
```

Note: When creating an ESDS file on an SFS server, use the SFS facility to limit the number of records that are in the file, because CICS cannot access records whose entry-sequence number cannot be packed into a 32-bit record byte address.

- A relative-record data set (RRDS) file:

CICS permits access to relative files only with fixed-length records. When defining such files, remember that the records include a field for the relative-record number, but this field is not directly visible to applications that use the CICS file control interface.

The file manager (SFS or DB2) permits relative-record files to have secondary indices. CICS cannot use these secondary indices, but their presence does not prevent the use of the file.

Assume that you have created a RRDS file named **level** whose primary index is **level_number** and whose secondary index is **description**. You can create a file definition that allows access with the primary, relative record number, and index, although CICS does not allow use of the secondary index. The primary file definition can be:

```
FD key      : LEVEL
BaseName    : level
IndexName   : level_number
```

Additionally you can create other File Definitions (FD), which refer to any of these files, and specify different values for some of the other attributes. You can, for example, create definitions with different RSL keys and different values for delete access status. Alternatively, you can also have two definitions that reference the same file name and index name but have different values for the recoverability status.

You can perform all the previous configuration tasks for the region while it is running by using the procedure that is described in “Configuring the File Definitions (FD) in the permanent and runtime databases.”

Note: SFS and DB2 permit more than one client to perform operations on the same file simultaneously. Therefore, you can add FD entries for the same file to more than one region, if the regions use suitable applications that have been designed to exploit this facility.

Configuring the File Definitions (FD) in the permanent and runtime databases

The following command examples show how to work with the File Definitions in the permanent database. By adding the **-B** (both permanent and runtime) option to the command, you also can include the definition in the runtime database.

- Adding a new FD:

This example shows how to add an FD entry for a file that CICS knows as **file1**, with an RSL key of **1**, for a file named **personnel**, which has an index name of **number**. Defaults are assumed for all other FD attributes.

```
cicsadd -c fd -r regionA -P file1 RSLKey=1 BaseName=personnel\
IndexName=number
```

If the region is running, and you want to add the definition to the runtime in addition to the permanent database, use the **-B** (both permanent and runtime) option, as shown:

```
cicsadd -c fd -r regionA -B file1 RSLKey=1 BaseName=personnel\
IndexName=number
```

- Changing an existing FD:

This example shows how to change an FD entry.

```
cicsupdate -c fd -r regionA -P file1 OpenStatus=open
```

- Deleting an FD entry:

This example shows how to delete an FD entry.

```
cicsdelete -c fd -r regionA -P file1
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- Viewing an FD entry:

This example shows how to inquire about an FD entry in the permanent database.

```
cicsget -c fd -r regionA file1
```

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

Working with FD in the runtime database

To work with the File Definitions (FD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change File Definitions (FD) while the region is running. Any changes that are made by using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions on the next restart. Use CEMT to do the following:

- View the runtime File Definitions (FD)
- View the location of runtime files
- View the status of a runtime file
- Change the runtime file status
- View or change runtime file-processing options
- View or change the base name of a runtime file
- View or change the index name of a runtime file
- View or change the file manager of a runtime file

For more information, see the *TXSeries for Multiplatforms Administration Reference*. To write CICS application programs to manage the status or operation of files in the runtime, see the *TXSeries for Multiplatforms Application Programming Reference*.

Adding files to the file manager by using Schema File Definitions (SCD)

For complete information about Schema File Definitions (SCD) and their attributes, see the *TXSeries for Multiplatforms Administration Reference*.

The following command examples show how to configure SCD for a file.

- Adding a new SCD entry:

This example shows how to add a new SCD entry. The permanent database option (-P) and the region name are not used because these are irrelevant to SCD entries.

```
cicsadd -c scd -f file.schema resourceName \  
    FileType=sequential \  
    FieldName1=happy \  
    FieldType1=unsignedInt16 \  
    FieldName2=sneezy \  
    FieldType2=unsignedInt16 \  
    FieldName3=grumpy \  
    FieldType3=unsignedInt16 \  
    FieldName4=dopey \  
    FieldType4=unsignedInt16
```

- Changing an existing SCD:
This example shows how to change an attribute in an SCD entry.
`cicsupdate -c scd -f file.schema resourceName FileType=clustered`
- Deleting an SCD entry:
This example shows how to delete an SCD entry.
`cicsdelete -c scd -f file.schema`
- Viewing an SCD entry:
This example shows how to view an SCD entry.
`cicsget -c scd -f file.schema`

On the AIX platform. See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region.

Setting up the data files on an SFS server with the `cicssfsimport` command

An SCD is a set of file and index specifications that can be applied to any file on any SFS server. In an SCD, you can specify:

- The type of file
- Field names, types, and lengths
- The maximum number of records for the file
- Primary and secondary index names
- Primary and secondary index descending fields
- File and secondary index preallocated pages
- File and secondary index volume names
- Variable length fields

The following procedure uses the `cicssfsimport` command to add files to the SFS logical volumes for CICS file management. It uses data that is specified in the SCD.

Before performing the procedure documented in this section, ensure that the following setup procedures are completed:

- Configuring the region; see Chapter 4, “Working with a file manager in a CICS region,” on page 23
- “Preparing the physical storage devices for the SFS server” on page 274 (if the SFS server is new or if the region cannot use existing logical volumes)
- “Defining an SFS server by using the `cicssfscreate` command” on page 275 (if the SFS server is new)
- “Starting an SFS server by using the `cicssfs` command” on page 276

Ensure that either the Region Definitions (RD) **DefaultFileServer** attribute or the File Definitions (FD) **FileServer** attribute is set to the name of the SFS server that is used for these files. For example, if you have been preparing hostB for this configuration, ensure that the **DefaultFileServer** or the **FileServer** attribute is set to `./cics/sfs/hostB`.

Attention: All files on the SFS logical volume are discarded when the SFS server is cold started. Each time that the SFS server is cold started, these files must be created again.

The following procedure uses the `cicssfsimport` command to add to the SFS logical volumes the files that are used for CICS file management:

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. If the SFS server has been set up to accept only authenticated RPCs, to apply a Schema File Definitions (SCD) to the SFS server, you need to be a member of the group **cics admin**.
3. Use the **cicssfsimport** command as described in the *TXSeries for Multiplatforms Administration Reference*. For example, to create the files FILE1 and FILE2 on a server hostB, using the entries from an SCD named SCHEMA1, use the command:

```
cicssfsimport -i SCHEMA1 -s ../cics/sfs/hostB \
-f FILE1 -x INDEX1 -x INDEX2 -x INDEX3 \
-f FILE2 -x INDEX4 -x INDEX5
```

With this command, FILE1 has indices INDEX1, INDEX2, and INDEX3, and FILE2 has indices INDEX4 and INDEX5.

CICS has facilities that protect the files against access from unauthorized users (as described in “Authorizing user access to CICS resources” on page 99).

Setting up the data files on DB2 with the **cicsdb2import** command

An SCD is a set of table and index specifications that can be applied to any table on any DB2 database. In an SCD, you can specify:

- The type of table
- Field names, types, and lengths
- The maximum number of records for the table
- Primary and secondary index names
- Primary and secondary index descending fields
- File and secondary index preallocated pages
- File and secondary index volume names
- Variable-length fields

The following procedure uses the **cicsdb2import** command to add tables to the DB2 database for CICS file management. It uses data that is specified in the SCD. Before you can add data files to the DB2 database, you must:

- Do the procedure that is given in “Preparing for DB2 file and queue management” on page 37.
 - Be logged in with a user ID that has DB2 administrative privileges.
1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
 2. Use the **cicsdb2import** command as described in the *TXSeries for Multiplatforms Administration Reference*. For example, to create tables for FILE1 and FILE2 on a database ALIAS2, using the entries from a schema file named SCHEMA1, you can use this command:

```
cicsdb2import -i SCHEMA1 -d ALIAS2 -u DB2USER -p xy93k1d \
-f FILE1 -x INDEX1 -x INDEX2 -x INDEX3 \
-f FILE2 -x INDEX4 -x INDEX5
```

In this command, FILE1 has indexes INDEX1, INDEX2, and INDEX3, and FILE2 has indexes INDEX4 and INDEX5. The DB2 user ID and password are passed to DB2 with this command.

Adding files to the file manager by using the diagnostic tools

You can also add files to the file manager by using diagnostic tools that are provided with CICS. “Setting up the data files on SFS using the `cicssdt` diagnostic tool” describes how to use the diagnostic tool for SFS files, and “Setting up the data files on DB2 with the `cicsddt` diagnostic tool” on page 140 describes how to use the diagnostic tool for DB2 files.

Setting up the data files on SFS using the `cicssdt` diagnostic tool

This procedure adds the user files to the SFS logical volumes for CICS file management by using the CICS SFS Diagnostic Tool (`cicssdt`). This tool provides a powerful interactive interface to the SFS server. Although several options are used with the `cicssdt` command, the **create** option is used most commonly because it is used to create new SFS files. You can also use the `cicssdt` command to do the following:

- List all the files on the SFS server
- Read records from a file on the SFS server
- Write records to a file on the SFS server
- Empty a file of all committed records
- Delete a file from the SFS server
- Add a secondary index to the SFS server file
- Delete a secondary index from the SFS server file
- Show detailed information about the SFS server file
- Transfer and convert a VSAM file to an SFS file
- Transfer an SFS file to a flat file
- Transfer a flat file to an SFS file

Attention: All files on the SFS logical volume are discarded when the SFS server is cold started. Each time that the SFS server is cold started, these files must be created again.

The procedures that are described in the following sections must be completed before you can add user files to the SFS server logical volumes:

- “Preparing the physical storage devices for the SFS server” on page 274 (if the SFS server is new or if the region cannot use existing logical volumes)
- Chapter 4, “Working with a file manager in a CICS region,” on page 23
- “Defining an SFS server by using the `cicssfscreate` command” on page 275 (if the SFS server is new)
- “Starting an SFS server by using the `cicssfs` command” on page 276

Ensure that either the Region Definitions (RD) **DefaultFileServer** attribute or the File Definitions (FD) **FileServer** attribute is set to the name of the SFS server that is used for these files. For example, if you have been preparing **hostB** for this configuration, ensure that the **DefaultFileServer** or **FileServer** attribute is set to `./cics/sfs/hostB`.

To use the `cicssdt` command to add to the SFS logical volumes the files that are used for CICS file management:

1. Log in with a user ID that has root privileges and is also a member of the group **cics**.
2. Use the `cicssdt` command as described in the *TXSeries for Multiplatforms Administration Reference* for each user file. For example, if you need to create a file named **FILE1** on SFS **hostB**, use the command:
`cicssdt -s hostB -c create FILE1`

When you use this command, you might be prompted to specify the following:

- The logical volume that is used for the files, such as **sfs_ShostB**.
- The file organization, such as **btreeClustered**.
- The names of the fields.
- The types of the fields, such as **byteArray** and **varLenByteArray**.
- The sizes of the fields.
- The maximum number of records.

Note: When the **cicssdt** utility is used to create a file that uses a relative-record data set (RRDS), and you want to place a limit on the number of records in the file, calculate the maximum to be the desired limit plus one.

- The primary index name.
- The index field names.
- The index field sequencing; for example, ascending.
- The number of pages to allocate.

CICS has facilities that protect the files against access from unauthorized users (described in “Authorizing user access to CICS resources” on page 99).

Setting up the data files on DB2 with the **cicsddt** diagnostic tool

This procedure adds files to the DB2 database for CICS file management by using the CICS DB2 Diagnostic Tool (**cicsddt**). This tool provides a powerful interactive interface to the DB2 database. Although several options are used with the **cicsddt** command, the **create** option is used most commonly because it is used to create new DB2 files. You can also use the **cicsddt** command to do the following:

- List all the files on the DB2 database
- Read records from a file on the DB2 database
- Write records to a file on the DB2 database
- Empty a file of all committed records
- Delete a file from the DB2 database
- Add a secondary index to the DB2 database file
- Delete a secondary index from the DB2 database file
- Show detailed information about the DB2 database file
- Transfer and convert a VSAM file to a DB2 file
- Transfer a DB2 file to a flat file
- Transfer a flat file to a DB2 file

The **cicsddt** command can also be used, in conjunction with the **cicssdt** command, to migrate files between SFS and DB2. See “Moving file management from SFS to DB2” on page 50.

Before using the procedure that is described in this section, ensure that the following procedures have been completed:

- Setting up DB2, as documented in “Preparing for DB2 file and queue management” on page 37.
- Logging in with a user ID that has DB2 administration privileges.

To use the **cicsddt** command to add to the DB2 database, files that are used for CICS file management :

1. Log with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.

2. Export the environment variables `CICSDB2CONF_CONNECT_USER` and `CICSDB2CONF_CONNECT_USING` to set the DB2 user ID and password.
3. Use the **cicsddt** command as described in the *TXSeries for Multiplatforms Administration Reference* for each data file. For example, if you need to create a file named `FILE1` on a DB2 database named `ALIAS`, use the command:

```
cicsddt -s ALIAS -c create FILE1
```

When this command is used, you might be prompted for the following:

- The database that is used for the files
- The file organization
- The names of the fields
- The types of the fields
- The sizes of the fields
- The maximum number of records
- The primary index name
- The index field names
- The index field sequencing; for example, ascending
- The number of pages to allocate

Chapter 11. Configuring CICS for use with relational databases

CICS transactions can access relational databases by including embedded SQL calls within the body of a CICS application program. Coordinated commitment and the recovery of transactions that include CICS and SQL calls are possible only with relational databases that support the X/Open XA interface.

Notes:

1. Releases of database products that are supported for this release of CICS are listed on the support page on the Web, which can be accessed from <http://www.ibm.com/cics/txseries/support>.
2. If you are using COBOL, see the same Web page to verify that the COBOL compiler that you are using is at the supported level.

CICS performs the role of a Transaction Manager in the X/Open distributed transaction processing (DTP) model (see the X/Open XA specification for more details). It can coordinate distributed transactions to relational databases that support the XA interface. The XA interface is not an application programming interface (API); it is a system-level interface between a transaction manager (TM) and a resource manager (RM) in the X/Open DTP model. Nevertheless, application programmers need to be aware that they are coding in an XA-enabled environment. For more information about application programming in such an environment, see the *TXSeries for Multiplatforms Application Programming Guide*.

"The CICS-provided database sample programs" tells you where you can find the sample database programs that are provided with CICS.

"Optimizing the single-phase commit protocol" on page 145 explores migration, backup, recovery, and problem determination for a single-phase commit implementation for data integrity.

"Using relational databases with CICS" on page 144 explains backup, recovery, and connectivity in XA-enabled environments.

"SQL database setup procedures" on page 149 details how to set up CICS for use with DB2, Informix, Oracle, and Sybase database products.

The CICS-provided database sample programs

CICS provides sample programs of XA-enabled and non-XA-enabled environments in the directory `$CICS/examples/RM_support`. It is strongly recommended that you use these sample programs to verify that your databases have been set up correctly *before* you attempt to use the databases in a production environment. On AIX, the instructions for XA-enabled and non-XA-enabled sample programs are in the file `CHEESE.README`. The `CHEESE.README` file describes an example database for a cheese company.

Using relational databases with CICS

CICS has an advanced load-based architecture under which transaction code is dynamically loaded into long-running application server processes. This architecture allows new transactions to be added to a running CICS system and eases maintenance because applications need not be relinked if an internal library change occurs.

To support this architecture, the vendors of XA-enabled relational databases supply shared reentrant versions of their libraries for use with CICS. These libraries are described in each vendor's documentation. Information about configuring CICS to work with various databases is provided in "SQL database setup procedures" on page 149.

Backup and recovery in XA-enabled environments

CICS supports statically and dynamically registering XA-enabled relational databases. If a database failure occurs, you must assess the impact that the failure can have on your CICS installation:

- In XA-enabled relational databases that use static registration, all CICS transactions that are in the region make XA calls. If the relational database has an irrecoverable failure, such as a disk-head failure or a remote link failure, CICS catches that failure and immediately abends the region. If you attempt to warm start the CICS region before the database is recovered, the region abends.
- In XA-enabled relational databases that use dynamic registration, only transactions that invoke SQL commands on an XA-enabled database make XA calls. If the relational database has an irrecoverable failure, CICS transactions that are running against the XA-enabled database fail. CICS catches the database failure only if a new CICS application server is started. CICS must be shut down before the database is brought back online after recovery. If you attempt an automatic start of the CICS region before the database is recovered, the region abends.

For these reasons, consider running another region that does not interact with the database. Then, if CICS is affected by a database shutdown, the other region can continue function shipping, transaction routing, and writing to the Structured File Server (SFS) server to maintain a limited CICS service.

If the relational database is likely to be unavailable for a long time and your system includes transactions that do not make database calls, you can perform a CICS cold start, but any transactions that were being processed at the time of the relational database failure are lost. If you perform a cold start, you must temporarily remove the Product Definitions (XAD) for the unavailable relational database from your CICS configuration.

If you wait until the relational database is fully recovered, a CICS warm start recovers all in-flight transactions from the time of the relational database failure.

Relay of catastrophic diagnostic information

A further implication of the close integration of XA-enabled relational databases is that CICS immediately abends the region if it cannot open a connection to the database. Catastrophic diagnostic information, such the failure of the initial connection to the database or subsequent irrecoverable failures in the database, is

relayed to the CICS systems administrator within the **console.nnnnnnn** and the system records files (**symrecs**). Further information about the nature of failures is available in the following locations:

- DB2: The operating system syslog
- Informix: No additional information
- Oracle: XA error log files
- Sybase: XA error log files

The application programmer is responsible for trapping SQL errors.

Optimizing the single-phase commit protocol

Some configurations, such as those with only one database resource, do not require the full data integrity that is provided by the two-phase commit protocol of the XA interface. A single-phase commit optimization enables higher performance for CICS configurations in which data integrity is not such a problem.

If a database failure occurs, CICS reports the possibility of inconsistent data from failed transactions. However, administrators need to check the consistency of data if multiple resource managers are involved in the failing transactions. If the database is the only resource manager that is participating in the transactions, the data is consistent. If the transactions make changes to other resources, such as CICS queues or files or other RDBMS tables, inconsistent data is possible. In such cases, you must resynchronize the various resources.

To enable the single-phase commit optimization, configure an appropriate Product Definitions (XAD) for a CICS region. For more details, see the section that discusses the specific vendor's database that you are using.

Note: Under the single-phase commit optimization, a CICS region supports, at most, one instance of any particular RDBMS. A CICS region cannot concurrently support a two-phase commit XA-enabled RDBMS with a single-phase commit optimized version of itself. A CICS region can concurrently support other XA-enabled relational databases, but full data integrity between the XA-enabled databases and the single-phase commit optimized RDBMS cannot be guaranteed.

To develop new applications for an RDBMS that is using the single-phase commit optimization, see the *TXSeries for Multiplatforms Application Programming Guide*, and follow the guidelines that are given for developing XA-enabled transactions.

Migration considerations for XA commit levels

Users of CICS who have an XA-enabled RDBMS can move from an XA two-phase commit environment to a single-phase commit environment by changing their Product Definitions (XAD) in the region database and performing a warm start. No application code changes are required, and users need not rebuild XA-enabled application code for this environment.

Users who want to move from a non-XA environment to a single-phase commit environment must configure a new Product Definitions (XAD) entry, but need to rework application code so that CICS performs transaction control, instead of EXEC SQL COMMIT or ROLLBACK statements.

Backup and recovery

An RDBMS that is using the single-phase commit optimization does not participate in CICS recovery. As mentioned previously, in some failure scenarios, you must manually resynchronize the database with other resource managers.

Because the single-phase commit optimization is still effectively an XA enablement, the RDBMS single-phase commit database is closely integrated with the CICS region. All CICS transactions that are in the region make SYNCPOINT processing calls to the database.

If the relational database has an irrecoverable failure, such as a disk-head crash, all transactions that are in the system fail and the region must be shut down. You cannot warm start the CICS region until the database has been recovered or the region abends. The administrative considerations that are discussed in “Backup and recovery in XA-enabled environments” on page 144 also apply to single-phase commit optimization.

Diagnosing problems

Error handling focuses on the propagation of the underlying SQL CONNECT, COMMIT, and ROLLBACK error codes and error messages to the administrator through the system console and the symptom records (**symrecs**) files.

Problem diagnosis requires RDBMS problem-determination documentation and skills. Further information about the nature of the failure can be found in RDBMS-specific error logs. See the relevant RDBMS enablement procedures for further details.

Enabling database connections in XA-enabled environments

When the region is started in an XA transaction environment, the CICS application server uses the XA open string to make an initial connection to the database. All CICS transactions run with either the default user **cics** or with a user that has been explicitly specified in the XA open string.

At a minimum, the default user and the user who is identified in the XA open string must be granted database and table access to the RM.

XA-enabled databases supply a structure called the XA switch, through which an XA Transaction Manager accesses the various XA functions that are supported by that database. CICS loads the XA switch structure at runtime. In CICS, this structure is called the *switch load file*.

You can build two types of switch load files: two-phase commit and single-phase commit. The two-phase commit switch load file is appropriate in those applications in which distributed Resource Managers are used; the single-phase commit version is appropriate in those applications in which only one Resource Manager is used. CICS supplies a source file and a makefile to build a switch load file for each supported RDBMS. The names of the source and make files and the instructions for creating the switch load file are given in “Using CICS with DB2” on page 150, “Using CICS with Informix” on page 153, “Using CICS with Oracle” on page 156, and “Using CICS with Sybase versions 11 and 12” on page 163.

CICS immediately abends the region if it cannot open a connection to an XA-enabled relational database.

Preparing the region for use with the RM

The environment variables that are required for the RM runtime are also required for the CICS region runtime. These values should be included in the region's environment file (`/var/cics_regions/regionName/environment`).

The region's environment is inherited from **root**'s shell when the region is started. You can add database environment information to the region's environment file (`/var/cics_regions/regionName/environment`). This action avoids connection errors that occur if the region is started from a shell that does not have the required database environment variables set.

Do not use \$ values in the region's environment file because these values are not expanded. Refer to the information that is given in the region's environment file prolog.

When adding the library path to the region's environment file, ensure that all the libraries that are required by the applications that run on this region are included. This action is necessary because the value for the library path that existed when the region was started is replaced with the value for the library path in the region's environment file. This is true for all values, such as PATH, NLSPATH, LANG, and so on.

Setting up Product Definitions (XAD)

The Product Definitions (XAD) are used to provide CICS with information about XA-compliant products that are accessed by CICS application programs. Use CICS commands to define an XA-compliant Resource Manager (RM) to CICS. For more information about the full syntax of CICS commands, see the *TXSeries for Multiplatforms Administration Reference*.

The following command examples show how to work with the XAD.

- Adding a new XAD:

This example shows how to add an XAD entry for a product named **xaproduct**.

```
cicsadd -c pd -r regionA xaproduct attribute=attributeValue ...
```

- Changing an existing XAD:

This example shows how to change an XAD entry in the permanent database.

```
cicsupdate -c xad -r regionA xaproduct attribute=attributevalue
```

- Deleting an XAD:

This example shows how to delete an XAD entry from the permanent database.

```
cicsdelete -c xad -r regionA xaproduct
```

- Viewing a PD entry:

This example shows how to view an XAD entry.

```
cicsget -c xad -r regionA sales1 xaproduct
```

Provide the RM-specific XA open string (the **XAOpen** attribute) and the location of the switch load file (the **SwitchLoadFile** attribute).

The **XAOpen** attribute of the XAD defines the text that is passed as an argument to the **xa_open** interface routine. You must provide the name of the database. Unless you also provide an operating system user ID and password, the region connects to the database under the default user name **cics**.

The **SwitchLoadFile** attribute of the XAD is used to specify the name of the switch load file that lists the names of the XA routines.

The syntax that each RDBMS uses to identify the XA open string and the procedure that is used to create the switch load file for each RDBMS is shown in “Using CICS with DB2” on page 150, “Using CICS with Informix” on page 153, “Using CICS with Oracle” on page 156, and “Using CICS with Sybase versions 11 and 12” on page 163. Additional XAD attributes are described in the *TXSeries for Multiplatforms Administration Reference*.

On the AIX platform. The System Management Interface Tool (SMIT) can also be used to add or change XAD. For more information, see Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259.

Notes:

1. An XAD is required for DB2 databases that are used for CICS queue and file management. However, the procedures that are described in “Administration of file and queue management on DB2” on page 31 creates the XAD for you.
2. Unlike all other resource classes, the XAD are loaded from the permanent database on a warm start. It is only when an emergency restart is performed that the XAD are loaded from the runtime database.
3. Adding environment variables to the file `/var/cics_regions/regionName/environment` can be required. “SQL database setup procedures” on page 149 describes when to do this.

CICS transactions can access relational database management systems (RDBMS) by including embedded SQL calls within the body of a CICS application program. For coordinated transaction commitment and recovery, however, the RDBMS must comply with the X/Open Distributed Transaction Processing (DTP) standard.

The X/Open DTP standard uses an application program (AP), resource manager (RM), and transaction manager (TM) model. In the CICS implementation, the CICS application program is the AP, the RDBMS is the RM, and CICS is the TM. Figure 10 shows the CICS implementation of this model.

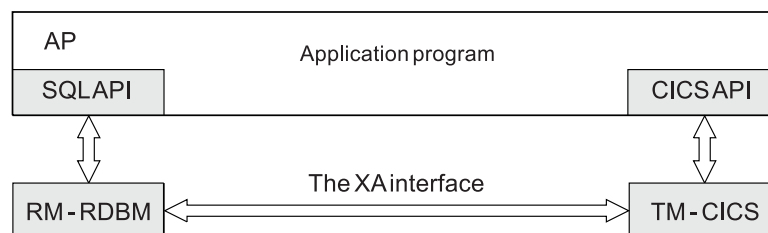


Figure 10. The CICS implementation of the AP, RM, TM model

Notes:

1. An XAD is required for DB2 databases that are used for CICS queue and file management. However, the procedures that are described in “Administration of file and queue management on DB2” on page 31 creates the XAD for you.
2. Unlike all other resource classes, the XAD are loaded from the permanent database on a warm start. It is only when an emergency restart is performed that the XAD are loaded from the runtime database.
3. Adding environment variables to the file `/var/cics_regions/regionName/environment` can be required. “SQL database setup procedures” on page 149 describes when to do this.

Enabling Micro Focus Server Express COBOL or ACUCOBOL-GT for use with the RM

The CICS COBOL runtime environment is required by CICS Application Servers when COBOL transactions are loaded. To create this environment, invoke the **cicsmkcobol** command (described in the *TXSeries for Multiplatforms Administration Reference*). If the COBOL runtime environment is specific to one region, invoke **cicsmkcobol** from that region's **bin** directory, **/var/cics_regions/regionName/bin**. If the COBOL runtime environment is to be shared by all regions on the current machine, invoke **cicsmkcobol** from the **\$CICS/bin** directory. The platform-specific locations are shown in Table 29:

Table 29. Location of the *cicsmkcobol* program

Platform	Directory
AIX	/usr/lpp/cics/bin
HP-UX	/opt/cics/bin
Solaris	/opt/cics/bin

For information about how to use Micro Focus Server Express COBOL with CICS, see Chapter 12, "Preparing CICS for use with Micro Focus Server Express COBOL," on page 175. For information about how to use ACUCOBOL-GT with CICS, see Chapter 13, "Preparing CICS for use with ACUCOBOL-GT," on page 177

SQL database setup procedures

This section describes how to set up the interface between CICS and the RDBMS. Procedures for interface setup are provided for the following RDBMS: DB2, Informix, Oracle, and Sybase. Instructions include support for enabling an XA transaction environment. If you want a non-XA transaction environment, do only the steps that affect preparation of the resource manager and the region. When you choose an XA transaction environment, you can use a single-phase commit or a two-phase commit protocol. If the resource manager is to function with the Micro Focus Server Express COBOL programming language, include the extra step that is required to build the COBOL runtime environment.

The overall flow of the setup procedure is the same for each RDBMS, although the specific commands within the overall flow can differ for each different RDBMS product. In addition, the prerequisite conditions can differ for each product. The following list summarizes the procedure that is used to set up the interface between CICS and the relational database manager for all supported resource management products:

- Step 1. Prepare the RM.
 - a. Enable connections between CICS and the RM.
 - b. Grant privileges required for use of the RM with CICS.
- Step 2. Build the switch load file (if an XA transaction environment is used).
- Step 3. Prepare the region.
 - a. Set up the region's environment.
 - b. Add the Product Definitions (XAD) to the region.
- Step 4. Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used).
- Step 5. Verify the connection.
- Step 6. Test the connection.

The following sections use this summary-level procedure to create a CICS interface for each supported relational database:

- For DB2, see “Using CICS with DB2.”
- For Informix, see “Using CICS with Informix” on page 153.
- For Oracle, see “Using CICS with Oracle” on page 156.
- For Sybase, see “Using CICS with Sybase versions 11 and 12” on page 163.

Using CICS with DB2

This procedure describes how to enable the region to connect to a DB2 RDBMS. It describes XA single-phase commit and two-phase commit connections, non-XA connections, and Micro Focus Server Express COBOL runtime requirements.

Prerequisite tasks and conditions

The DB2 instance and database must be defined.

Procedure for setting up the CICS interface to DB2

This section describes how to enable transactions to be executed from within CICS in an XA environment that can use either the single-phase commit or two-phase commit protocol. You can configure a non-XA enabled environment by doing only the steps that are associated with the DB2 RDBMS and the region preparation. You can enable the use of Micro Focus Server Express COBOL by including the steps that are associated with the COBOL product.

Step 1. Prepare the DB2 RM

- a. Enable connections between CICS and the DB2 RM. Optionally, on Open Systems platforms when an XA transaction environment is used, you can set the TP_MON_NAME database configuration parameter to **cics** to improve performance. The command is:

```
update database manager configuration using \
tp_mon_name cics
```

- b. Grant privileges that are required for use of the DB2 resource manager with CICS:

- Grant connection privileges as required by your security setup to the user names **root**, **cics**, and the user name that has been explicitly specified in the XA open string. The user **cics** must have DB2 administration group access to the database in which the CICS queues and files reside.

At a minimum, this user requires the following privileges (expressed as an SQL statement):

```
grant bindad on database to user userName
grant select on table sysibm.sysindexes to user userName
```

where the *userName* is either the default user **cics** or the user who is specified in the XA open string (see Step 3b on page 151).

Step 2. Build the DB2 switch load file (if an XA transaction environment is used).

- Two switch load files are supplied with CICS:
 - **\$CICS/bin/cics1pcdb2** for single-phase commit
 - **\$CICS/bin/cicsxadb2** for two-phase commit

It is recommended that you use these supplied files, as described in step 3b on page 151. However, if library or symbol-access errors occur because of changes in the DB2 library (such as when a DB2 maintenance upgrade is applied), use the following CICS-supplied source and makefiles to rebuild the DB2 switch load file:

- **db21pc.sqc** and **db21pc.mk** for single-phase commit
 - **db2xa.c** and **db2xa.mk** for two-phase commit
- a. Go to the CICS directory that contains the source and makefile for the switch load file that you want to rebuild. For example:
\$CICS/src/examples/RM_support/DB2
 - b. Copy the switch load file source and makefile into the CICS **bin** directory:

On the AIX platform:

```
cp db2xa* /usr/lpp/cics/bin
or
cp db21pc* /usr/lpp/cics/bin
```

On the HP-UX and Solaris platforms:

```
cp db2xa* /opt/cics/bin
or
cp db21pc* /opt/cics/bin
```

Note: If your switch load file is intended for exclusive use by a region, copy the makefile to the region's **bin** directory:

/var/cics_regions/regionName/bin

- c. Change to the directory into which you copied the makefile in step 2b. Verify that the pathname in the makefile for the **db2.o** file is accurate. If non-standard paths are used, the pathnames in the makefile might need to be modified.
- d. Use the makefile to build the switch load file:
make -f db2xa.mk
or
make -f db21pc.mk

Step 3. Prepare the region for DB2.

- a. Set up the region's environment.

On the AIX platform. The region's environment file must include the variable that indicates the name of your DB2 instance, **DB2INSTANCE=DB2instanceName**, where *DB2instanceName* is the name of your DB2 instance.

On the HP-UX platform. The library path variable, **\$DB2DIR/lib**, must be included in the **SHLIB_PATH** library path setting. If you are using Micro Focus Server Express COBOL, the name of your DB2 instance, **DB2INSTANCE=DB2instanceName**, must be included in your region's environment file.

On the Solaris platform. The library path variable, **\$DB2DIR/lib**, must be included in the **LD_LIBRARY_PATH** library path setting. If you are using Micro Focus Server Express COBOL, the name of your DB2 instance, **DB2INSTANCE=DB2instanceName**, must be included in your region's environment file. In addition, if you are running any COBOL applications, the region's environment file must include the variable **LD_PRELOAD=libdb2.so**.

- b. Add the Product Definitions (XAD) to the region.

The XA open string in DB2 uses the following format:
databaseName[,userName,password], where:

- *databaseName* is the name of the database. If you have explicitly cataloged an alias name after database creation, use the alias name.

- *userName* is an operating system user ID. This is optional. Specify a user ID if you do not want the region to connect to the database as the user **cics**.
- *passWord* is the password for the *userName* user ID.

Use the **cicsadd** command to add the XAD to the region. For example, depending on your operating system:

On the AIX platform:

```
cicsadd -c xad -r regionName DB2XAD \
    ResourceDescription="XA Connection for DB2" \
    XAOpen="cicsDB2database" \
    SwitchLoadFile="/usr/lpp/cics/bin/cicsdb2xa"
```

On the HP-UX and the Solaris platforms:

```
cicsadd -c xad -r regionName DB2XAD \
    ResourceDescription="XA Connection for DB2" \
    XAOpen="cicsDB2database" \
    SwitchLoadFile="/opt/cics/bin/cicsdb2xa"

cicsadd -c xad -r regionName DB2XAD \
    ResourceDescription="XA Connection for DB2" \
    XAOpen="cicsDB2database" \
    SwitchLoadFile="/opt/ibm/cics/bin/cicsdb2xa"
```

where the value for **SwitchLoadFile** indicates either **cicsdb2xa** for two-phase commit or **cics1pcdb2** for single-phase commit. See step 2 on page 150 for the names of your built files.

Step 4. Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used).

To build a CICS COBOL runtime that includes the DB2 environment:

- Verify that your shell includes all environment variables that are required for COBOL, DB2, and CICS application development.
- From the shared libraries, extract the required object file (**sqlgmf.o**) that is needed by the COBOL runtime for CICS and DB2. To extract and link the object file, enter the commands:

```
cd $DB2DIR/lib
ar -vx libdb2gmf.a
ln -s $DB2DIR/lib/sqlgmf.o /usr/lib/sqlgmf.o
```

- Change to the directory in which you want to build the COBOL runtime: either the region's **bin** directory or the CICS system's **bin** directory. Table 29 on page 149 shows the appropriate directory for each supported platform.
- Enter the following command from the region's **bin** directory, depending on your operating system:

On the AIX platform:

```
cicsmkcobol $DB2DIR/lib/db2.o /usr/lib/sqlgmf.o
```

On the HP-UX and the Solaris platforms:

```
cicsmkcobol -L$DB2DIR/lib $DB2DIR/lib/sqlgmf.o -ldb2
```

Step 5. Verify the connection.

CICS logs XA messages in the region's console file. To determine whether the region can connect to the database:

- Start your region. If this is a newly created region, follow the steps that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:
- ```
ciscp start region regionNameStartType=cold
```

- b. In a second window, use the following command to view the console messages:

```
cicstail -r regionName
```

If XA problems occur, check for additional information in the system log. For more information, see the *DB2: Problem Determination Guide*.

**Step 6. Test the connection.**

You can use the CICS-supplied examples to test fully XA and non-XA connections. Use the examples to see how to code and compile your CICS/SQL application programs. See “The CICS-provided database sample programs” on page 143 for the location of the examples directory on each platform.

## Using CICS with Informix

This section describes how to enable the region to connect to an Informix database. It describes XA single-phase commit and two-phase commit connections, non-XA connections, and Micro Focus Server Express COBOL runtime requirements.

### Prerequisite tasks and conditions

For the region to connect to an Informix database, the following prerequisites must be met:

- Informix environment: If the Informix server and client environments reside on the same machine, the Informix products must be installed in the sequence that is specified by the Informix documentation.
- Informix server environment: An Informix server environment must exist and be accessible. See Informix documentation for a description of how to install the Informix Online Workgroup Server.
- Informix client environment: An Informix client environment must exist. The Informix client environment consists of the Informix ESQL (C or COBOL or both).

If the database server and clients reside on the same machine, install these Informix products before you install the Informix server. See Informix documentation for an explanation of how to install the Informix ESQL products and for a fuller description of how to configure the Informix database server.

- Informix Server Configuration: No specific configuration is required to operate the Informix Online Workgroup Server in an XA environment, although unbuffered transaction logging must be defined for the database.
- Informix Client Configuration: If Micro Focus Server Express COBOL transactions are to be used, ensure that the COBOL environment is correctly set up as indicated by Informix documentation.

### Procedure for setting up the CICS interface to Informix

This section describes how to enable transactions to be executed from within CICS in an XA environment that can use either the single-phase commit or two-phase commit protocol. You can enable a non-XA enabled environment by performing only the steps that are associated with the Informix resource manager and the region preparation. You can enable the use of Micro Focus Server Express COBOL by including the steps that are associated with the COBOL product.

**Step 1. Prepare the Informix resource manager.**

- a. Enable connections between CICS and the Informix resource manager.  
Databases that are accessed by transaction managers must be created with unbuffered logging. No other modification is required for use of Informix with CICS.

- b. Grant privileges that are required for use of the Informix resource manager with CICS.

When an XA-enabled region is started, the CICS Application Server makes an initial connection to the database that is identified by the XA open string, and uses the server that is identified by the variable `INFORMIXSERVER`. The Application Servers are owned by user `cics`, which must be given access to the database where processing is to take place.

You can give access to user `cics` by invoking the `dbaccess` Informix ISQL facility as the Informix database administrator, selecting the user database, and running the following SQL command:

```
grant resource to cics;
```

Also, by granting resource privileges to the user ID `root`, you can enable offline access to the CICS queues and files through the `dbaccess` command.

- Step 2. Build the Informix switch load file (if an XA transaction environment is used).

To build the single-phase commit or the two-phase commit switch load file, compile the source and make these files:

- `informix_xa.c` and `informix_xa.mk` for two-phase commit
  - `inform1pc.c` and `inform1pc.mk` for single-phase commit
- a. Go to the CICS directory that contains the source and makefile for the switch load file that you want to rebuild. For example:  
`$CICS/src/examples/RM_support/Informix`
  - b. Copy the switch load file source and makefile into the CICS `bin` directory:

**On the AIX platform.**

```
cp informix* /usr/lpp/cics/bin
or
cp inform1pc* /usr/lpp/cics/bin
```

**On the HP-UX and the Solaris platforms.**

```
cp informix* /opt/cics/bin
or
cp inform1pc* /opt/cics/bin
```

**Note:** If your switch load file is intended for exclusive use by a region, copy the makefile to the region's `bin` directory:  
`/var/cics_regions/regionName/bin`

- c. Change to the directory into which you copied the makefile in step 2b. Verify that the pathname in the makefile for the `inform*.o` file is accurate. If nonstandard paths are used, the pathnames in the makefile might need to be modified.
- d. Use the makefile to build the switch load file:  

```
make -f informix_xa.mk
or
make -f inform1pc.mk
```

- Step 3. Prepare the region for Informix.

- a. Set up the region environment for Informix.

**On the AIX platform.** The region's environment file must include the values for the variables `INFORMIXDIR` and `INFORMIXSERVER`.

**On the HP-UX and Solaris platforms.** The region's environment file must include the values for the variables `INFORMIXDIR` and



INFORMIXSERVER. Also, the library path variables **\$INFORMIXDIR/lib** and **\$INFORMIXDIR/lib/esql** must be included in the library path setting.

- b. Add the Product Definitions (XAD) to the region.

For Informix 7.2, the **XAOpen** attribute has the syntax *dataBaseName*, where *dataBaseName* is the name of the Informix database.

An example of using the **cicsadd** command to add the product definition to the region follows:

```
cicsadd -c xad -r InfXAD \
 ResourceDescription="XA Connection for Informix" \
 XAOpen="databaseName" \
 SwitchLoadFile="switchLoadFileName"
```

where *databaseName* is the name of the database, and *switchLoadFileName* is the name of the switch load file that you created in step 2 on page 154. When supplying the **XAOPEN** attribute, the **cics** user ID and password are implied, unless you supply a specific user ID and password.

- Step 4. Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used).

Build a CICS COBOL runtime as follows:

- a. Verify that your shell includes all environment variables, Informix libraries, and objects that are required for COBOL, Informix, and the CICS application environment.
- b. Change to the directory in which you want to build the COBOL runtime: either the region's **bin** directory or the CICS system's **bin** directory. Table 29 on page 149 shows the appropriate directory for each supported platform.
- c. **On the AIX platform.** Enter the command:

```
cicsmkcobol -L $INFORMIXDIR/lib/esql \
 $INFORMIXDIR/lib/esql/libcicsshr.o
```

**On the HP-UX and Solaris platforms.** This process is not supported.

- Step 5. Verify the connection.

CICS logs XA messages in the region's console file. To determine whether the region can connect to the Resource Manager, start your region and check the console messages.

- a. Start your region. If this is a newly created region, follow the steps that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:

```
cicsstart -c regionName StartType=cold
```

- b. In a second window, use this command to view console messages:

```
cicstail -r regionName
```

Check the messages when they are logged to the console. The first XA messages are generated when the CICS Application Servers are started. If XA problems occur, obtain additional information with the Informix utilities **onstat** and **onlog**. (See the Informix documentation for more information.)

- Step 6. Run a test transaction.

The CICS-supplied examples provide a way to test XA and non-XA connections. See "The CICS-provided database sample programs" on page 143 for the location of the examples directory on each platform.



## Using CICS with Oracle

This section describes how to enable the region to connect to an Oracle database. It describes non-XA connections, XA two-phase commit with and without dynamic registration, XA single-phase commit connections, and Micro Focus Server Express COBOL runtime requirements.

### Prerequisite tasks and conditions

Ensure that the following prerequisites are met:

- The required Oracle software is installed as described in the instructions that are given in the relevant Oracle installation documentation:
  - Oracle Server (RDBMS)
  - SQL\*Net
  - TCP/IP Protocol Adapter
  - Oracle XA library (required for XA support only)
  - Pro\*C (required for C applications only)
  - Pro\*COBOL (required for COBOL applications only)
  - SQL\*Plus
- An Oracle server is running.

### Procedure for setting up the CICS interface to Oracle

This section describes how to enable transactions to be executed from within CICS in an XA environment that can use either the single-phase commit or two-phase commit protocol with and without dynamic registration. You can configure a non-XA enabled environment by performing only the steps that are associated with resource and region preparation. You can enable use with Micro Focus Server Express COBOL by including the steps that enable COBOL.

#### Prepare the Oracle resource manager:

1. Enable connections between CICS and the Oracle resource manager.  
No extra setup is required.
2. Grant privileges that are required for use of the Oracle resource manager with CICS.

Select permissions must be granted on the **dba\_pending\_transactions** table to any Oracle user that will be specified in an XA open string or to **public**. To do this, start a **sqlplus** session as the system administrator and issue a command that has the following syntax:

```
grant select on dba_pending_transactions to scott;
```

where **scott** is the Oracle connect user.

The **dba\_pending\_transactions** view is created when Oracle is installed.

3. If you are in a non-XA enabled environment, and using the BEQUEATH protocol, enable the **Turning Off UNIX Signal Handling** option in the Oracle Network Client Configuration profile. See the *Oracle Network Manager Administration Guide* for information about how to set this option. If you do not do this, your programs might encounter inconsistent behavior; for more information about this, refer to APAR II13428 and Oracle's MetaLink note 74839.1.
4. To take advantage of Oracle's dynamic registration, build the switchload file with a **libclntsh.a** that contains CICS libraries. Before you build the switchload, modify Oracle script **genclntsh**, which is found in directory **\$ORACLE\_HOME/bin**. Change the **OLIBS** string from:

```
OLIBS="${LCLIENT} ${LSQL} ${LNETWORK} ${LCLIENT} ${LCOMMON} ${LGENERIC}
${LMM} -bI:${ORACLE_HOME}/rdbms/lib/xa.imp ${LNLSRTL} ${LCORE}${LNLSRTL}
${LCORE} ${LNLSRTL} ${LWTC} ${LNETWORK} ${LCLIENT} -l${COMMON} ${LGENERIC}
${LTRACE} ${LNLSRTL} ${LCORE} ${LNLSRTL} ${LCORE} ${LNLSRTL} ${LPLSQL}"
```

to

```
OLIBS="${LCLIENT} ${LSQL} ${LNETWORK} ${LCLIENT} ${LCOMMON} ${LGENERIC}
${LMM} -bI:${ORACLE_HOME}/rdbms/lib/xa.imp -L/usr/lpp/cics/lib -lEncServer
${LNLSRTL} ${LCORE} ${LNLSRTL} ${LCORE} ${LNLSRTL} ${LWTC} ${LNETWORK}
${LCLIENT} -l${COMMON} ${LGENERIC} ${LTRACE} ${LNLSRTL} ${LCORE} ${LNLSRTL}
${LCORE} ${LNLSRTL} ${LPLSQL}"
```

You must then run the `genclntsh` script to build the `libclntsh.a` file that contains the correct libraries.

### Build the Oracle switch load file (if an XA transaction environment is used):

CICS supplies the following source and makefiles for Oracle switch load files:

- **oraclexa\_dyn.c** and **oraclexa\_dyn.mk** for two-phase commit with dynamic registration
- **oraclexa\_sta.c** and **oraclexa\_sta.mk** for two-phase commit without dynamic registration
- **oracle1pc.c** and **oracle1pc.mk** for single-phase commit

Build the single-phase or two-phase commit switch load file with or without dynamic registration by compiling the appropriate source file as described in one of the following sections:

*Build a two-phase commit dynamic switch load file for Oracle 9i with dynamic registration:*

1. Verify that your shell includes all the environment variables that are required for Oracle, and the CICS application environment.
2. Ensure that the Oracle client libraries are available on the machine on which the dynamic registration switch load file needs to be created.
3. On the machine on which the dynamic registration switch load file needs to be created, shut down any programs that use the Oracle client library, including regions that have XA connections to Oracle.
4. On AIX, unload the shared libraries at the kernel level. Enter the command:  
`/usr/sbin/slibclean`
5. Log in as the user **oracle**; for example, enter the command:  
`su - oracle`
6. Change to the `$ORACLE_HOME/bin` directory.
7. Make a backup copy of the `genclntsh` script.
8. Make the following changes to the `genclntsh` script:
  - On AIX:
    - a. Replace the value of the variable `XAONDY` with the following:  
`XAONDY="-L/usr/lpp/cics/lib -lEncServer"`
  - On HP-UX or Solaris:
    - a. Uncomment the line that reads `set -xa`.
    - b. Replace the string `${ORACLE_HOME}/rdbms/lib/xaondy.o` with the following string  
`-Lcics_install_directory/lib -lEncServer`
9. Back up the Oracle client library if it exists, as follows:

- On AIX:  
mv \$ORACLE\_HOME/lib32/libcIntsh.a \$ORACLE\_HOME/lib32/libcIntsh.a.bak
  - On HP-UX:  
mv \$ORACLE\_HOME/lib32/libcIntsh.sl \$ORACLE\_HOME/lib32/libcIntsh.sl.bak
  - On Solaris (Oracle 9i 64-bit installation):  
mv \$ORACLE\_HOME/lib32/libcIntsh.so \$ORACLE\_HOME/lib32/libcIntsh.so.bak
  - On Solaris (Oracle 9i 32-bit installation):  
mv \$ORACLE\_HOME/lib/libcIntsh.so \$ORACLE\_HOME/lib/libcIntsh.so.bak
10. Run the genclntsh script. Enter one of the following commands:
- On AIX:  
genclntsh -32
  - On HP-UX:  
genclntsh lib32
  - On Solaris (Oracle 9i 64-bit installation):  
genclntsh lib32
  - On Solaris (Oracle 9i 32-bit installation):  
genclntsh
11. Change to the Oracle lib directory:
- Oracle 9i 64-bit installation:  
cd \$ORACLE\_HOME/lib32
  - Oracle 9i 32-bit installation:  
cd \$ORACLE\_HOME/lib
12. Ensure that the Oracle client library includes the required CICS library. Enter one of the following commands:
- On AIX:  
/usr/bin/dump -Hv libcIntsh.a
  - On HP-UX:  
/usr/bin/ldd libcIntsh.sl.9.0
  - On Solaris:  
/usr/bin/ldd libcIntsh.so.9.0
- Ensure that the output contains a listing for the following library:
- On AIX:  
libEncServer.a
  - On HP-UX:  
libEncServer.sl
  - On Solaris:  
libEncServer.so
13. Log in as the user **root**.
14. Copy oraclexa\_dyn.c and oraclexa\_dyn.mk from CICS examples directory to either the CICS bin directory or the CICS region's bin directory, as required.

**Note:** If your switch load file is intended for exclusive use by a CICS region, copy the files to the region's bin directory:

/var/cics\_regions/*regionName*/bin

- On AIX:  
cd /usr/lpp/cics/bin  
cp /usr/lpp/cics/src/examples/RM\_support/Oracle/oraclexa\_dyn\* .
- On HP-UX and Solaris:

```
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/oraclexa_dyn* .
```

15. Build the switch load file. Enter one of the following commands:

- On AIX and HP-UX:
 

```
make -f oraclexa_dyn.mk oracle9
```
- On Solaris with Oracle 9i (64-bit):
 

```
make -f oraclexa_dyn.mk oracle9_64
```
- On Solaris with Oracle 9i (32-bit):
 

```
make -f oraclexa_dyn.mk oracle9_32
```

*Build a two-phase commit dynamic switch load file for Oracle 8.1.7 with dynamic registration:*

1. Verify that your shell includes all the environment variables that are required for Oracle and the CICS application environment.
2. Ensure that the Oracle client libraries are available on the machine on which the dynamic registration switch load file needs to be created.
3. On the machine on which the dynamic registration switch load file needs to be created, shut down any programs that use the Oracle client library, including regions that have XA connections to Oracle.
4. [AIX] Unload the shared libraries at the kernel level. Enter the command:
 

```
/usr/sbin/slibclean
```
5. Log in as the user **oracle**; for example, enter the command:
 

```
su - oracle
```
6. Change to the \$ORACLE\_HOME/bin directory
7. Make a backup copy of the genclntsh script.
8. Make the following edits to the genclntsh script:
  - a. Uncomment the line that reads `set -xa`
  - b. Replace the string `${ORACLE_HOME}/rdbms/lib/xaondy.o` with the following string
 

```
-Lcics_install_dir/lib -lEncServer
```
9. Run the genclntsh script.
10. Change to the \$ORACLE\_HOME/lib32 directory.
11. Ensure that the Oracle client library includes the required CICS library. Enter the command:
 

```
[AIX] $ /usr/bin/dump -H libclntsh.a
[HP-UX] /usr/bin/ldd libclntsh.sl
[Solaris] /usr/bin/ldd libclntsh.so
```

Ensure that the output contains a listing for the following library:

```
[AIX] libEncServer.a
[HP-UX] libEncServer.sl
[Solaris] libEncServer.so
```
12. Log in as the user **root**.
13. Copy oraclexa\_dyn.c and oraclexa\_dyn.mk from CICS examples directory to either the CICS bin directory or the CICS region's bin directory, as required.

**Note:** If your switch load file is intended for exclusive use by a CICS region, copy the files to the region's bin directory:

```
/var/cics_regions/regionName/bin
```

- [AIX]
 

```
cd /usr/lpp/cics/bin
cp /usr/lpp/cics/src/examples/RM_support/Oracle/oraclexa_dyn* .
```

- **[HP-UX or Solaris]**

```
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/oraclexa_dyn* .
```

14. Build the switch load file. Enter the command:

```
make -f oraclexa_dyn.mk oracle8
```

*Build a single-phase or two-phase commit static switch load file (without dynamic registration):*

1. Verify that your shell includes all the environment variables that are required for Oracle and the CICS application environment.
2. Copy oraclexa\_sta.c and oraclexa\_sta.mk from the CICS examples directory to either the CICS bin directory or the CICS region's bin directory, as required.

**Note:** If your switch load file is intended for exclusive use by a CICS region, copy the files to the region's bin directory:  
/var/cics\_regions/regionName/bin

- **[AIX]**

```
cd /usr/lpp/cics/bin
cp /usr/lpp/cics/src/examples/RM_support/Oracle/oraclexa_sta*.*
```

- **[HP-UX or Solaris]**

```
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/oraclexa_sta.* .
```

3. Change to the directory into which you copied the makefile in step 2.

4. Build the switch load file. Enter one of the following commands:

- On Sun Solaris:

- Two-phase commit static switch load file for Oracle 9i (64-bit)  
make -f oraclexa\_sta.mk oracle9\_64
- Two-phase commit static switch load file for Oracle 9i (32-bit)  
make -f oraclexa\_sta.mk oracle9\_32
- One-phase commit switch load file for Oracle 9i (64-bit)  
make -f oracle1pc.mk oracle9\_64
- One-phase commit switch load file for Oracle 9i (32-bit)  
make -f oracle1pc.mk oracle9\_32

- On AIX and HP-UX:

- Two-phase commit static switch load file for Oracle 9i  
make -f oraclexa\_sta.mk oracle
- One-phase commit switch load file for Oracle 9i  
make -f oracle1pc.mk oracle

- On all platforms:

- One-phase commit switch load file for Oracle 8i  
make -f oracle1pc.mk oracle8

**Prepare the region for Oracle:**

**Step 1.** Set up the region's environment.

**On AIX.** Include ORACLE\_HOME and ORACLE\_SID in the region's environment file.

**On HP-UX or Solaris.** Include ORACLE\_HOME and ORACLE\_SID in the region's environment file, and include \$ORACLE\_HOME/lib in the library path.

**Step 2.** Add the Product Definitions (XAD) to the region.

The following XAD attributes in each stanza are of particular importance:

### **XAOpen**

This attribute defines the XA open string. The string follows:

`Oracle_XA +Acc=P/username/password+SesTm=sessionTimeLimit`

The user name and password can be null if "P/" is a valid user name-password pair.

The value for `SesTm=sessionTimeLimit` indicates the maximum time that is allowed in a transaction for the commit or rollback of the transaction, or between one service and the next service. The unit of time for this value is given in seconds. A value of 0 indicates no limit, but use of this value is discouraged. The setting `SesTm=15` indicates that the session idle time limit is 15 seconds.

Refer to your Oracle documentation for a detailed description of all fields that can be used in the XA open string.

An example of an XA open string for Oracle is:

`Oracle_XA+Acc=P/scott/tiger+SesTm=35+LogDir=/tmp+DbgFl=1`

This string causes a CICS application server process (**cicsas**) to connect to the database, as follows:

- As user **scott** with password **tiger** (`Acc=P/scott/tiger`)
- With inactive transactions that are aborted after 35 seconds (`SesTm=35`)
- With log files that are created in the **/tmp** directory (`LogDir=/tmp`)
- With XA procedure calls and returns that are logged in the log file (`DbgFl=1`)

### **SwitchLoadFile**

This attribute is a string that defines the name of the switch load file object. Instructions that explain how to build an appropriate switch load file are given in step "Build the Oracle switch load file (if an XA transaction environment is used)" on page 157.

For example; if your switch load file is called `oraclexa_dyn` and it is in the region's bin directory, set the `SwitchLoadFile` attribute to `oraclexa_dyn`.

### **XAClose**

This attribute defines the XA close string. It is set to a null string "".

The **cicsadd** command is used to add an XAD stanza to the XAD stanza file. For example, with the following command, you can add a stanza with the attributes as described above.

```
cicsadd -c xad -r regionName Oracle_XA \
 ResourceDescription="Oracle XA product definition" \
 XAOpen="Oracle_XA+Acc=P/scott/tiger+SesTm=35+LogDir=/tmp+DbgFl=1" \
 XAClose="" \
 SwitchLoadFile="oraclexa_dyn" $cicsadd -c xad -r
```

**Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used):** Build a COBOL runtime that includes the Oracle libraries as follows:

- Step 1. Verify that your shell includes all the environment variables that are required for COBOL, Oracle, and the CICS application environment.
- Step 2. Change to the directory in which you want to build the COBOL runtime: either the region's **bin** directory or the CICS system's **bin** directory. Table 29 on page 149 shows the appropriate directory for each supported platform.
- Step 3. Enter the following command, depending on your operating system:

**On the AIX platform:**

```
cicsmkcobol $ORACLE_HOME/precomp/lib/cobsqintf.o \
 $ORACLE_HOME/lib/libcintsh.a
```

**On HP-UX or Solaris:**

```
cicsmkcobol -L $ORACLE_HOME/lib -lcintsh -lsql\
 $ORACLE_HOME/precomp/lib/cobsqintf.o
```

**Verify and test the connection:**

- Step 1. Verify the connection.

CICS logs XA messages in the region's console file. Determine whether the region can connect to the Resource Manager as follows:

- a. Start your region. If this is a newly created region, follow the directions that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:

```
cicscp start regionregionName StartType=cold
```

When you start the region on Solaris, you might receive the following warning message:

```
ld.so.1: oracletxora: warning: /usr/lib/libc.so: open failed:
illegal insecure pathname
```

You can ignore these warnings because they have no effect on your system. You can also ignore these warnings if you receive them when you run transactions in the region.

- b. In a second window, tail the console messages. Enter the command:

```
cicstail -r
```

Check the messages when they are logged to the console. The first XA messages are generated when the CICS application servers are started. If XA problems occur, check for additional information in the Oracle trace file. (See step 2 on page 160.)

- Step 2. Test the connection.

The CICS-supplied examples provide an easy way to test XA and non-XA connections. See "The CICS-provided database sample programs" on page 143 for the location of the examples directory on each platform.

## Building Pro\*C and Pro\*COBOL programs on Oracle 9i

Oracle release 9i supports 32-bit and 64-bit clients. Currently, CICS client programs must be built under 32-bit mode only. Therefore with Oracle 9i (64-bit), you can use only 32-bit versions of Pro\*C and Pro\*COBOL components.

The following table shows the naming convention that must be used to build Pro\*C and Pro\*COBOL programs under the CICS and Oracle 9i environment.

|                                  | Oracle 9i (64-bit)           | Oracle 9i (32-bit)         |
|----------------------------------|------------------------------|----------------------------|
| Pro*Cobol precompiler executable | \$(ORACLE_HOME)/bin/procob32 | \$(ORACLE_HOME)/bin/procob |



|                               |                                    |                                  |
|-------------------------------|------------------------------------|----------------------------------|
| Pro*C precompiler executable  | \$(ORACLE_HOME)/bin/proc           | \$(ORACLE_HOME)/bin/proc         |
| AIX Oracle client library     | \$(ORACLE_HOME)/lib32/libclntsh.a  | \$(ORACLE_HOME)/lib/libclntsh.a  |
| AIX LIBPATH                   | \$(ORACLE_HOME)/lib32              | \$(ORACLE_HOME)/lib              |
| HP-UX Oracle client library   | \$(ORACLE_HOME)/lib32/libclntsh.sl | \$(ORACLE_HOME)/lib/libclntsh.sl |
| HP-UX SHLIB_PATH              | \$(ORACLE_HOME)/lib32              | \$(ORACLE_HOME)/lib              |
| Solaris Oracle client library | \$(ORACLE_HOME)/lib32/libclntsh.so | \$(ORACLE_HOME)/lib/libclntsh.so |
| Solaris LD_LIBRARY_PATH       | \$(ORACLE_HOME)/lib32              | \$(ORACLE_HOME)/lib              |

## Using CICS with Sybase versions 11 and 12

This procedure describes how to enable the region to connect to a Sybase Logical Resource Manager (LRM). It describes XA single-phase commit and two-phase commit connections, non-XA connections, and Micro Focus Server Express COBOL runtime requirements. For instructions about how to use Sybase version 11 with CICS, see “Prerequisite tasks and conditions for use of Sybase version 11 with CICS” and “Procedure for setting up the CICS interface to Sybase version 11” on page 164. For instructions about how to use Sybase version 12 with CICS, see “Prerequisite tasks and conditions for use of Sybase version 12 with CICS” on page 168 and “Procedure for setting up the CICS interface to Sybase version 12” on page 169.

**On the HP-UX platform.** When Sybase version 11 or version 12 is used with CICS on the HP-UX platform, you must consider NLS character configuration requirements. “NLS issues for Sybase with CICS for the HP-UX platform” on page 173 provides instructions for preparing the locale configuration for use on an HP-UX machine.

**Note:** *Configuring Sybase(R) Adaptive Server Enterprise for UNIX Platforms* describes how to enable an XA connection for CICS.

### Prerequisite tasks and conditions for use of Sybase version 11 with CICS

Add the following Sybase version 11 environment variable definitions to the **.profile** file:

```
export DSQUERY=sybase_server_name
export PATH=$SYBASE/install:$PATH
export XCONFIGFILE=$SYBASE/xa_config
```

Ensure that the following prerequisites are met:

- Sybase server environment: A Sybase server environment must exist and be accessible. See Sybase documentation for information about how to install the Sybase Adaptive Server.
- Sybase XA Interface: Use one of the following:
  - To run XA-enabled transactions, a Sybase XA Interface environment is required. The following Sybase products must be installed in the sequence listed:
    1. The Sybase Embedded SQL (C and/or COBOL)
    2. The Sybase XA Interface/C Developers kit

3. The Sybase XA-library for CICS
  - To run non-XA-enabled transactions, a Sybase XA Interface environment is required. The following Sybase version 11 products must be installed in the sequence listed:
    1. The Sybase Embedded SQL (C and/or COBOL)
    2. The Sybase XA Interface/C Developers kit
 See the Sybase documentation for information about how to install Sybase version 11.
- Sybase server configuration: Start your Sybase SQL server as described in the instructions that are given in the Sybase documentation.
  - Verify that a Sybase system administrator database (**sybssystemdb**) is configured to provide XA functions. Refer to descriptions of **xa\_load.sh** that are given in the Sybase documentation.
  - To prevent errors when you start your region, ensure that Sybase allows enough connections to accommodate the total number of CICS application servers that can require access at any given time.
  - If a Sybase configuration file (**ocs.cfg**) is used, ensure that a server level entry is included in that file. Otherwise, regions that have XA connections fail to start.
- Sybase XA Interface configuration: Use one of the following:
  - To run XA-enabled transactions, update the Sybase XA configuration file **\$SYBASE/xa\_config**. The Sybase XA configuration file identifies the Sybase server with a logical resource manager (LRM) name. The LRM name is used in the XA open string of the CICS XAD Product Definition. The capabilities and properties of an LRM can also be specified in the Sybase XA configuration file.
  - The Sybase server that is identified in the XA configuration file must also be identified in the **\$SYBASE/interfaces** file.
  - To run non-XA transactions, the **\$SYBASE/interfaces** file must identify the Sybase server to which it is connected.
  - **On the AIX platform.** To ensure that the loader has no problems with the Sybase XA Interface library files, run the **\$SYBASE/sample/xalibrary/CICS/mkshlibs.sh** script.

For more information, see the Sybase documentation.

## Procedure for setting up the CICS interface to Sybase version 11

This section describes how to enable transactions that are to be executed from within CICS in an XA environment that can use either the single-phase commit or two-phase commit protocol. You can configure a non-XA enabled environment by performing only the steps that are associated with the Sybase LRM version 11 and the region preparation. You can enable the use of Micro Focus Server Express COBOL by including the steps that are associated with the COBOL product.

### Step 1. Prepare the Sybase LRM version 11.

- a. To enable connections between CICS and the Sybase LRM version 11, edit the **xa\_config** file to add the following information:

For both XA and non-XA implementations:

```
logfile=/databases/sybase/xa_log
traceflags=xa
properties=CS_LOGIN_TIMEOUT=35
lrm=connection_1
server=sybase_server
```

When the connection that you are enabling is an XA implementation, also add the following information to the **xa\_config** file:

```
lrm=connection_1
server=sybase_server
```

An entry is required in the **\$SYBASE/xa\_config** file for each connection of the logical resource manager. In this example, **connection\_1** is the name of the LRM and **sybase\_server** is the name of the server. The LRM **connection\_1** is used in the **-N** parameter of the **XAOpen** attribute of the XAD. It is also used in the transaction program EXEC SQL SET CONNECTION statement to identify this connection to the Sybase server named **SYBASE**.

It is possible to have only one connection per logical unit of work; however, multiple connections can be used in a CICS task. In this case, each connection must have an XAD entry defined to the region.

See the Sybase documentation for more information.

Also, if an **ocs.cfg** file is used, an entry for the server **SYBASE** is required in the **ocs.cfg** file.

**Note:** The region fails to start if the **ocs.cfg** file exists and a server-level entry is not included in it.

To configure specifications at other levels for the connection, see the Sybase documentation.

- b. Grant privileges that are required for use of the Sybase LRM version 11 with CICS.

When an XA-enabled region is started, the CICS application server uses the XA open string to make an initial connection to the LRM. For Sybase, the XA open string identifies the user name and password for which all CICS XA transactions are performed. It is important to ensure that the XA CICS user has access to the Sybase XA-stored procedures.

**On the AIX platform.** The examples that are referred to in step 6 on page 168 use a database called **cheese**. The following **isql** commands set up an XA CICS user as **cicsid** with a home database **cheese** and a password **cicspw**. The **isql** commands are run from within a Sybase system administrator's session:

```
sp_addlogin cicsid, cicspw, cheese
go
use cheese
go
sp_adduser cicsid
go
quit
```

**On the HP-UX and Solaris platforms.** The examples that are referred to in step 6 on page 168 use a database called **cicstest**. The following **isql** commands set up an XA CICS user as **cicsid** with a home database **cicstest** and a password **cicspw**. The **isql** commands are run from within a Sybase system administrator's session:

```
sp_addlogin cicsid, cicspw, cicstest
go
use cicstest
go
sp_adduser cicsid
go
quit
```

**Step 2.** Build the switch load file (if an XA transaction environment is used).

The switch load file is required only in an XA-enabled environment.

**On the AIX platform.** The source for the switch load file is in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make-f sybasexa.mk
cd /usr/lpp/cics/bin
cp /usr/lpp/cics/src/examples/RM_support/Sybase/sybasexa.*
```

**On the HP-UX version 10 platform.** The source for the switch load file is in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make-f sybasexallhp10.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexallhp10.*
```

**On the HP-UX version 11 platform.** The source for the switch load file is in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make-f sybasexallhp11.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexallhp11.*
```

**On the Solaris platforms.** The source for the switch load file is in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make-f sybasexall.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexall.*
```

**Note:** If your switch load file is intended for exclusive use by a region, copy the makefile to the region's **bin** directory:  
**/var/cics\_regions/regionName/bin**

**Step 3.** Prepare the region.

- a. Set up the region's environment for Sybase version 11.

The region's environment file must include the following variables:

**SYBASE=\$SYBASE**

In addition, the **\$SYBASE/lib** path must be included in the library path.

- b. Add the Product Definitions (XAD) to the region.

The XA specification allows each RM vendor to define an open string, which specifies the parameters that the XA interface uses to connect to the RM.

The **XAOpen** attribute of the XAD has the following syntax:

**-U userName -P password -N LRMname -L logFile [-T traceLevel]**

where:

- *userName* and *passWord* identify the connection to the Sybase version 11 database. Both must refer to a valid Sybase user account. Sybase user accounts are created by the Sybase system administrator, which uses the **sp\_addlogin** and **sp\_adduser** stored procedures (see *Sybase Adaptive Server Enterprise Reference Manual*).
- *LRMname* is the name of the LRM, as defined in the Sybase XA configuration file **\$SYBASE/xa\_config**. The LRM name is identified in the XA transaction with the command 'EXEC SQL SET CONNECTION LRMname'.
- *logFile* is the full pathname of the file to which tracing information is written. Because this log file is needed for error messaging, Sybase must have write access for it.
- *traceLevel* is the level of trace that is written to the log file. If trace is wanted, set *traceLevel* to **xa** to give entry- and exit-point tracing for XA procedures.

See the Sybase documentation for other valid settings. The **SwitchLoadFile** attribute of the XAD identifies the location of the switch load file that was built in step 2 on page 165. The example identifies '**sybasexa11**' as the switch load file. No path is given, so '**sybasexa11**' is expected to exist in the region's **bin** directory. For example, the **cicsadd** command to add a Product Definition (XAD) to the region is as follows:

```
cicsadd -c xad -r regionName SybaseXA \
 ResourceDescription="XA Connection for Sybase"\
 XAOpen="-Ucicsid -Pcicspw -NlrmName -L/tmp/lrmXA.log"\
 XASerialize="multiple_association" \
 SwitchLoadFile="sybasexa11"
```

Because the Sybase resource manager is thread-aware, make XA connections with **multiple\_association** mode for CICS with Sybase version 12.0 or later. To do this, you must set **XASerialize="multiple\_association"** in the XAD definition as shown above.

- Step 4. Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used).

Before you invoke **cicsmkcobol**, do the following:

- a. Change to the directory where your COBOL runtime will reside.
- b. Extract the objects from the Sybase COBOL library:  

```
ar -x $SYBASE/lib/libcobct_r.a
```
- c. Set up an environment variable that specifies the objects that are extracted from the Sybase COBOL library. The following example shows how to do this from the command line:

```
COBOJS='ar -t $SYBASE/lib/libcobct_r.a'
```

where **'** is a single backquote.

- d. Determine which Sybase libraries are required in the COBOL runtime environment.
  - For non-XA-enabled COBOL transactions, the following Sybase XA Interface libraries are required, depending on your operating system:  
**On the AIX platform:**  
**-libxlibct\_r.so -libcs\_r.so -libintl\_r.so -libtcl\_r.so -libcomn\_r.so**  
**On the HP-UX and the Solaris platforms:**

-libcs\_r.so -libct\_r.so -libctl\_r.so -libcomn\_r.so -libintr\_r.so

- For XA-enabled COBOL transactions, the following Sybase XA Interface libraries are required, depending on your operating system:

**On the AIX platform:**

-lxa -lct\_r.so -lcs\_r.so -lintl\_r.so -ltcl\_r.so -lcomn\_r.so -lsarpc -lm

**On the HP-UX platform:**

-lxa -lcs\_r.so -lct\_r.so -ltcl\_r.so -libcomn\_r.so -libintr\_r.so

**On the Solaris platform:**

-lxa -lcs\_r.so -lct\_r.so -lintr\_r.so -lsarpc

For a non-XA connection to Sybase, exclude the **-lxa** library.

It can be useful to set up an environment variable that specifies the Sybase libraries that are to be included in the CICS COBOL runtime environment. The following example shows how to do this from the command line:

```
SYBLIBS=ListOfLibraries
```

where *ListOfLibraries* matches the list that is shown for your operating system.

- e. Build the COBOL runtime. If the environment variables have been set as described above, enter the command:

```
cicsmkcobol -L $SYBASE/lib $COB0BJS $SYBLIBS
```

**Note:** Define SYBASE as in step 3a on page 166.

**Step 5.** Verify the connection.

CICS logs XA messages in the region's console file. To determine whether the region can connect to the LRM, start your region and check the console messages.

- Start your region. If this is a newly created region, do the steps that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:

```
cicscp start region regionName StartType=cold
```

- In a second window, tail the console messages. Enter the command:

```
cicstail -r regionName
```

The examples can also be used to see how to code and compile your CICS/SQL application programs. See the "The CICS-provided database sample programs" on page 143 for information about the location the example files on each platform.

**Step 6.** Run a test transaction.

The CICS-supplied examples provide an excellent way to test XA and non-XA connections fully. These examples are in the locations that are indicated in "The CICS-provided database sample programs" on page 143 for each platform.

## Prerequisite tasks and conditions for use of Sybase version 12 with CICS

Add the following Sybase version 12 environment variable definitions to the **.profile** file:

```
export DSQUERY=sybase_server_name
export PATH=$SYBASE_12_0/install:$PATHexport XCONFIGFILE=$SYBASE/xa_config
```



Ensure that the following prerequisites are met:

- Sybase Server environment: A Sybase server environment must exist and be accessible. See Sybase documentation for information about how to install the Sybase Adaptive Server.
  - Sybase XA Interface: Use one of the following:
    - To run XA-enabled transactions, a Sybase XA Interface environment is required. The following Sybase version 12 products must be installed in the sequence listed:
      1. The Sybase Embedded SQL (C and/or COBOL)
      2. The Sybase XA Interface/C Developers kit
      3. The Sybase XA library for CICS
    - To run non-XA-enabled transactions, a Sybase XA Interface environment is required. The following Sybase version 12 products must be installed in the sequence listed:
      1. The Sybase Embedded SQL (C and/or COBOL)
      2. The Sybase XA Interface/C Developers kit
- See the Sybase documentation for information about how to install Sybase version 12.

**Note:** For Sybase version 12, the Java JRE must be installed on the local host. See your Sybase documentation for additional information.

- Sybase XA Interface: Use one of the following:
  - To run XA-enabled transactions, update the Sybase XA configuration file **\$SYBASE/xa\_config**. The Sybase XA configuration file identifies the Sybase server with a logical Resource Manager (LRM) name. The LRM name is used in the XA open string of the CICS XAD Product Definition. The capabilities and properties of an LRM can also be specified in the Sybase XA configuration file.
  - The Sybase server that is identified in the XA configuration file must also be identified in the **\$SYBASE/interfaces** file.
  - To run non-XA transactions, the **\$SYBASE/interfaces** file must identify the Sybase server to which it is connected.

For more information, see the Sybase documentation.

## Procedure for setting up the CICS interface to Sybase version 12

This section describes how to enable transactions that are to be executed from within CICS in an XA environment that can use either the single-phase commit or two-phase commit protocol. A non-XA enabled environment can be configured by performing only the steps that are associated with the Sybase LRM version 12 and the region preparation. You can enable the use of Micro Focus Server Express COBOL by including the steps that are associated with the COBOL product.

**Step 1.** Prepare the Sybase LRM version 12.

- a. To enable connections between CICS and the Sybase LRM, edit the **xa\_config** file to add the following information:

```
[ALL]
logfile=/databases/sybase12/xa_log
traceflags=xa
properties=CS_LOGIN_TIMEOUT=35
[XA]
lrm=connection_1
server=sybase_server
```



An entry is required in the `$SYBASE/xa_config` file for each connection of the logical resource manager. In this example, `connection_1` is the name of the LRM and `sybase_server` is the name of the server. The LRM `connection_1` is used in the `-N` parameter of the `XAOpen` attribute of the XAD. It is also used in the transaction program `EXEC SQL SET CONNECTION` statement to identify this connection to the Sybase server named `SYBASE`.

It is possible to have only one connection per logical unit of work; however, multiple connections can exist in a CICS task. In this case, each connection must have an XAD entry defined to the region.

- b. Grant privileges that are required for use of the Sybase LRM version 12 with CICS.

When an XA-enabled region is started, the CICS application server uses the XA open string to make an initial connection to the LRM. For Sybase, the XA open string identifies the user name and password for which all CICS XA transactions are performed. It is important to ensure that the XA CICS user has access to the Sybase XA-stored procedures.

**On the AIX platform.** The examples that are referred to in step 6 on page 173 use a database called `cheese`. The following `isql` commands set up an XA CICS user as `cicsid` with a home database `cheese` and a password `cicspw`. The `isql` commands are run from within a Sybase system administrator's session:

```
sp_addlogin cicsid, cicspw, cheese
go
use cheese
go
sp_adduser cicsid
go
quit
```

**On the HP-UX and Solaris platforms.** The examples that are referred to in step 6 on page 173 use a database called `cicstest`. The following `isql` commands set up an XA CICS user as `cicsid` with a home database `cicstest` and a password `cicspw`. The `isql` commands are run from within a Sybase system administrator's session:

```
sp_addlogin cicsid, cicspw, cicstest
go
use cicstest
go
sp_adduser cicsid
go
quit
```

- c. **On the AIX platform.** The following entries are used to configure the server for Distributed Transaction Management (DTM):

```
sp_configure "enable DTM", 1
go
sp_configure 'enable xact coordination', 1
go
sp_configure 'txn to pss ratio', 25
go
sp_role "grant", dtm_tm_role, cheese
go
```

**Note:** The `'txn to pss ratio'` represents the number of available transactions, and the recommended minimum setting is the number of CICS application servers that are connected to the database at any one time.

**On the HP-UX and Solaris platforms.** The following entries are used to configure the server for XA support:

```
sp_configure "enable DTM", 1
go
sp_configure 'enable xact coordination', 1
go
sp_configure 'txn to pss ratio', 25
go
sp_role "grant", dtm_tm_role, cicstest
go
```

**Note:** The 'txn to pss ratio' represents the number of available transactions, and the recommended minimum setting is the number of CICS application servers that are connected to the database at any one time.

See Sybase documentation for more information.

**Step 2.** Build the switch load file (if an XA transaction environment is used).

The switch load file is required only in an XA-enabled environment.

**On the AIX platform.** The source for the switch load file is located in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make -f sybasexa.mk
cd /usr/lpp/cics/bin
cp /usr/lpp/cics/src/examples/RM_support/Sybase/sybasexa.*
```

**On the HP-UX version 10 platform.** The source for the switch load file is located in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make -f sybasexa12hp10.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexa12hp10.*
```

**On the HP-UX version 11 platform.** The source for the switch load file is located in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase** and then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make -f sybasexa12hp11.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexa12hp11.*
```

**On the Solaris platform.** The source for the switch load file is located in the **\$CICS/src/examples/RM\_support/Sybase** directory under the CICS root directory. To build the switch load file, build the binary file in **\$CICS/src/examples/RM\_support/Sybase**, then copy it to the CICS **bin** directory:

```
cd $CICS/src/examples/RM_support/Sybase
make -f sybasexa12.mk
cd /opt/cics/bin
cp /opt/cics/src/examples/RM_support/Sybase/sybasexa12.*
```

**Note:** If your switch load file is intended for exclusive use by a region, copy the makefile to the region's **bin** directory:  
`/var/cics_regions/regionName/bin`

**Step 3.** Prepare the region.

- a. Set up the region's environment for Sybase.

The region's environment file must include the following variables:

```
SYBASE=$SYBASE
XACONFIGFILE=$XACONFIGFILE
SYBASE_OCS=$SYBASE_OCS
SYBASE_ASE=$SYBASE_ASE
SYBASE_SYSAM=$SYBASE_SYSAM
```

**On the HP-UX and the Solaris platforms:** In addition, the **\$SYBASE/lib** path must be included in the library path.

- b. Add the Product Definitions (XAD) to the region.

The XA specification allows each RM vendor to define an open string, which specifies the parameters that the XA interface uses to connect to the RM.

The **XAOpen** attribute of the XAD has the following syntax:

```
-U userName -P passWord -N LRMname -L logFile [-T traceLevel]
```

where

- *userName* and *passWord* identify the connection to the Sybase database. Both must refer to a valid Sybase user account. Sybase user accounts are created by the Sybase system administrator, using the **sp\_addlogin** and **sp\_adduser** stored procedures (see *Sybase Adaptive Server Enterprise Reference Manual*).
- *LRMname* is the name of the LRM, as defined in the Sybase XA configuration file **\$SYBASE/xa\_config**. The LRM name is identified in the XA transaction with the command 'EXEC SQL SET CONNECTION *LRMname*'.
- *logFile* is the full pathname of the file to which tracing information is written. Because this log file is needed for error messaging, Sybase must have write access for it.
- *traceLevel* is the level of trace written to the log file. If trace is wanted, setting this to **xa** gives entry- and exit-point tracing for XA procedures.

See Sybase documentation for other valid settings. The **SwitchLoadFile** attribute of the XAD is used to identify the location of the switch load file that was built in step 2 on page 171. The example identifies '**sybasexa12hp11**' as the switch load file. No path is given, so '**sybasexa12hp11**' is expected to exist in the region's **bin** directory. For example, the **cicsadd** command to add a Product Definition (XAD) to the region is given below:

```
cicsadd -c xad -r regionName SybaseXA \
ResourceDescription="XA Connection for Sybase" \
XAOpen="-Ucicsid -Pcicspw -Nconnection_1 -L/tmp/lrmXA.log" \
SwitchLoadFile="sybasexa12hp11"
```

**Step 4.** Build the Micro Focus Server Express COBOL runtime (if COBOL is to be used).

Before you invoke **cicsmkcobol**, perform the following activities:

- a. Change to the directory in which your COBOL runtime will reside.  
b. Extract the objects from the Sybase COBOL library:

```
ar -x $SYBASE/$SYBASE_OCS/lib/libcobct_r.a
```

- c. Set up an environment variable that specifies the objects that are extracted from the Sybase COBOL library. The following example shows how to do this from the command line:

```
COBLIBS="-L$SYBASE/$SYBASE_OCS/lib \
-lxatm -lct_r.so -lcs_r.so -ltcl_r.so -lcomm_r.so -lintl_r \
`ar -t $SYBASE/$SYBASE_OCS/lib/libcobct_r.a`"
```

where ` is a single backquote.

- d. Build the COBOL runtime. If the environment variables have been set as described above, enter the command:

```
ar -x $SYBLIB/libcobct_r.a
cicsmkcobol $COBLIBS
```

#### Step 5. Verify the connection.

CICS logs XA messages in the region's console file. To determine whether the region can connect to the LRM, start your region and check the console messages.

- Start your region. If this is a newly created region, follow the steps that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:

```
cicscp start region regionName StartType=cold
```

- In a second window, tail the console messages. Enter the command:

```
cicstail -r regionName
```

The examples can also be used to see how to code and compile your CICS/SQL application programs. See the "The CICS-provided database sample programs" on page 143 for information about the location of the example files on each platform.

#### Step 6. Run a test transaction.

The CICS-supplied examples provide an excellent way to fully test XA and non-XA connections. These examples are in the locations that are indicated in "The CICS-provided database sample programs" on page 143 for each platform.

### NLS issues for Sybase with CICS for the HP-UX platform

**On the HP-UX platform.** You must ensure that the character set that the CICS locale uses is installed on the Sybase server. For example, if the CICS region is running in the en\_US.iso88591 locale, the ISO 8859-1 (Latin-1) character set must be installed on the Sybase server. You can add character sets to the existing server by using the Sybase command **sybinit**, as shown in "Procedure for adding a character set to the Sybase server on HP-UX." In addition, the locale must be added to Sybase's locale mapping file, as shown in "Adding an entry to Sybase's locale mapping file on HP-UX" on page 174.

#### Procedure for adding a character set to the Sybase server on HP-UX:

1. **On the HP-UX platform.** Start the Sybase initialization utility by using the following command at the command line:  

```
% sybinit
```
2. From the displayed menu, choose option 3, **Configure a server product.**
3. From the displayed menu, choose option 1, **SQL Server.**
4. From the display menu, choose option 2, **Configure an existing SQL server.**
5. From the displayed menu, choose option 7, **Configure Character Sets.**

6. A list of available character sets is displayed; enter the number of the character set of your choice and press <Return>.
7. Exit from the Sybase initialization utility on your HP-UX platform.

**Adding an entry to Sybase's locale mapping file on HP-UX:** On the HP-UX platform, Sybase's locale mapping file is located in the `/opt/sybase/locales/locales.dat` directory:

Add the locale name that is used by the CICS region to the file. The highlighted line in the following example shows the addition of the `en_US.iso88591` locale:

```
locale = n-computer, us_english, roman8
locale = C, us_english, roman8
locale = american, us_english, roman8
locale = american.iso88591, us_english, iso_1
locale = en_US.iso88591, us_english, iso_1
locale = english.iso88591, us_english, iso_1
locale = english, us_english, roman8
```

---

## Configuring ACUCOBOL-GT<sup>®</sup> for use with relational databases

On Solaris and HP-UX platforms, you need to specify the ACUCOBOL-GT environment variables `SHARED_LIBRARY_LIST` in the region's environment file to point to the appropriate database client library for XA as well as non-XA transactions. On the AIX platform, this should be specified for non-XA transactions only. The following examples show how you can specify the environment variables in the region's environment file for DB2:

On AIX:

```
SHARED_LIBRARY_LIST="DB2_dir/lib/libdb2.a(shr.o)"
```

On HP-UX:

```
SHARED_LIBRARY_LIST="DB2_dir/lib/libdb2.sl"
```

On Solaris:

```
SHARED_LIBRARY_LIST="DB2_dir/lib/libdb2.so"
```

Where `DB2_dir` is the directory in which DB2 is installed.

On non-AIX platforms, you can specify only the base name of the library if DB2 is in your shell's executable and library search path.

The following example show how you can specify the environment variables in the region's environment file for Oracle on AIX:

```
SHARED_LIBRARY_LIST="oracle-home/lib32/libc1ntsh.a(shr.o)"
```

where `oracle-home` is the directory in which Oracle is installed.

In general, you can list the database shared libraries in the configuration variable, `SHARED_LIBRARY_LIST`. If, however, you need to use database functions that are not packaged in a shared library, you must relink the ACUCOBOL-GT runtime library (for example, `libruncbl.so` on AIX), by using the `makerun` utility of ACUCOBOL, and build a new CICS language method file (`cicsprACUCOB`). See the ACUCOBOL-GT documentation for more information about relinking the runtime and using shared libraries. Whenever possible, use the configuration variable to specify shared libraries.

---

## Chapter 12. Preparing CICS for use with Micro Focus Server Express COBOL

---

### Setting up the Micro Focus Server Express COBOL runtime environment

The *TXSeries for Multiplatforms Application Programming Guide* describes how to use the supported compilers. Some configuration is required for Micro Focus Server Express COBOL.

#### When to use **cicsmkcobol**

If you install or upgrade your Micro Focus Server Express COBOL product, you must use the **cicsmkcobol** command. This command creates the Micro Focus Server Express COBOL language method file. To be used by CICS, you must place the COBOL language method file in either the **/var/cics\_regions/regionName/bin** directory, or the CICS system **/bin** directory. Table 30 shows the location of the CICS system **/bin** directory for each supported platform:

Table 30. Location of the CICS system **/bin** directory

| Platform | Directory                |
|----------|--------------------------|
| AIX      | <b>/usr/lpp/cics/bin</b> |
| HP-UX    | <b>/opt/cics/bin</b>     |
| Solaris  | <b>/opt/cics/bin</b>     |

You must use the name **cicsprCOBOL**. Versions of the file that are placed into **/var/cics\_regions/regionName/bin** take precedence over the file that is placed into the CICS system **/bin** directory. This allows you to have different COBOL language method files for each region.

See the *TXSeries for Multiplatforms Administration Reference* and *TXSeries for Multiplatforms Application Programming Guide* for more information.

---

### Micro Focus Server Express COBOL programs and relational databases

CICS can use Micro Focus Server Express COBOL application programs with DB2, Informix, Oracle, and Sybase relational database products. For information about how to configure these connections, see “SQL database setup procedures” on page 149.

If you want to run a mixed workload of Micro Focus Server Express COBOL and C transactions and use CICS on HP-UX with DB2 as resource manager, ensure that you have installed the Micro Focus Server Express COBOL with the ODBC support

---

## Micro Focus Server Express COBOL license management

Micro Focus Server Express COBOL programs require licenses. To enable this, the Micro Focus License Management Facility must be running on your system.

On AIX platforms, add the following three lines to the **/etc/mflmrscript** file:

```
COBDIR=/usr/lpp/cobol
export COBDIR
echo "6,9" | $COBDIR/aslmf/aptrack > /dev/null
```

On Solaris and HP-UX platforms, add the following three lines to the **/etc/mflmrscript** file:

```
COBDIR=/opt/lib/cobol
export COBDIR
echo "6,9" | $COBDIR/aslmf/aptrack > /dev/null
```

For more information, see the Micro Focus documentation.



---

## Chapter 13. Preparing CICS for use with ACUCOBOL-GT

---

### Setting up the ACUCOBOL-GT runtime environment

The *TXSeries for Multiplatforms Application Programming Guide* describes how to use the supported compilers. Some configuration is required for ACUCOBOL-GT.

#### When to relink the ACUCOBOL-GT runtime and create cicsprACUCOB

If you need to customize the ACUCOBOL-GT environment to use third-party software that is not distributed as a shared library, or if you modify one of more of the .c files in \$ACUCOBOL/lib, you must relink the ACUCOBOL-GT runtime library (libruncl.so) by using the **makerun** utility of ACUCOBOL-GT.

For third-party software that is distributed in a shared library, you do not need to relink the runtime. Instead, you can list the shared libraries that you want to link in the ACUCOBOL-GT configuration variable, `SHARED_LIBRARY_LIST`, in your region's environment file. See the ACUCOBOL-GT documentation for more information about relinking the runtime and using shared libraries. Whenever possible, specify shared libraries by using a configuration variable.

The language method file for ACUCOBOL-GT, which is called `cicsprACUCOB`, is provided prebuilt with CICS, and is present in the CICS system bin directory. You can, however, have a different `cicsprACUCOB` for each region, in the region's bin directory. The region-specific file takes precedence over the file that is in the CICS system bin.

If you upgrade ACUCOBOL-GT to a new version, it is recommended that you build a new `cicsprACUCOB`. You can do this as shown in the following example on AIX:

1. Back up any old `cicsprACUCOB` in `/usr/lpp/cics/bin`.
2. Change directory to `/opt/acu/lib`.
3. Enter the commands:

```
make -f makefile.txs
cp -p cicsprACUCOB /usr/lpp/cics/bin
```

You do not need to rebuild a new `cicsprACUCOB` if you customize the ACUCOBOL-GT runtime, because the ACUCOBOL-GT runtime is contained in a shared library (libruncl.so).

---

### ACUCOBOL-GT programs and relational databases

CICS can use ACUCOBOL-GT application programs with DB2, Oracle, Sybase, and Informix relational database products. For information about how to configure these connections, see "SQL database setup procedures" on page 149.

---

## ACUCOBOL-GT license management

ACUCOBOL-GT programs require licenses. To enable this, the ACUCOBOL-GT acushare program must be running on your system. If the acushare daemon is not running, CICS/ACUCOBOL-GT transactions will not run. To start the daemon, enter the command:

```
acushare -start
```

Optionally, you can set these environment variables for the system, the region, or in the .profile for the **cics** user:

- **PATH**, which contains the ACUCOBOL-GT bin directory (for example, /opt/acu/bin)
- **LIBPATH**, which contains the ACUCOBOL-GT lib directory (for example, /opt/acu/lib/usr/lib:/lib)

Before starting acushare, you will need to install the appropriate ACUCOBOL-GT licenses to run acucobol programs inside a CICS environment. To install the license:

1. Concatenate the ACUCOBOL-GT runtime license files runchl.alc and runchl.1lc from the \$ACUCOBOLbin/directory into /etc/license.acu.
2. From the \$ACUCOBOL/bin directory, type:  

```
cat runchl.alc runchl.1lc > /etc/license.acu
```
3. Start/Restart acushare after performing the above step.

If you are using license files for other Acucorp products, such as Acu4GL, those license files must be concatenated together in license.acu file.

For more information about the license management utility, see the *ACUCOBOL-GT Getting Started Guide*.

---

## Chapter 14. Working with WebSphere MQ

This chapter describes how to make CICS work with WebSphere MQ.

---

### Installation

These are the WebSphere MQ product components that are to be installed:

- WebSphere MQ Server
- WebSphere MQ Samples
- WebSphere MQ Messages

**Note:** The installation of components "WebSphere MQ Server" and "WebSphere MQ Messages" is mandatory.

For information about how to plan for and install WebSphere MQ, see the *WebSphere MQ: Quick Beginnings* book for your platform. For information about using CICS applications with WebSphere MQ, see the *WebSphere MQ Application Programming Guide*.

---

### Building your application

#### On the AIX platform.

CICS WebSphere MQ applications that are written in C require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
CCFLAGS="-I$MQM_HOME/inc"
LDLAGS="-L$MQM_HOME/lib -lmqm_r"
```

CICS WebSphere MQ applications that are written in COBOL (other than ACUCOBOL-GT) require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
COBCPY="$MQM_HOME/inc"
COBLIBS="-L$MQM_HOME/lib $MQM_HOME/lib/libmqmcbt.o -lmqm_r"
```

CICS WebSphere MQ applications that are written in ACUCOBOL-GT require that you set the following configuration variable (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
COPYPATH="$MQM_HOME/inc"
```

#### On the HP-UX platform.

CICS WebSphere MQ applications that are written in C require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
CCFLAGS="-I$MQM_HOME/inc"
LDLAGS="-L$MQM_HOME/lib -lmqm_r"
```

CICS WebSphere MQ applications that are written in COBOL require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
COBCPY="$MQM_HOME/inc"
COBLIBS="-L$MQM_HOME/lib $MQM_HOME/lib/libmqmcbt.o -lmqm_r"
```

### On the Solaris platform.

CICS WebSphere MQ applications that are written in C require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
CCFLAGS="-I$MQM_HOME/inc"
LDLAGS="-L$MQM_HOME/lib -lmqm -lmqmcs_d -lmqmzse"
```

CICS WebSphere MQ applications that are written in COBOL require the following compiler options (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
COBCPY="$MQM_HOME/inc"
COBLIBS="-L$MQM_HOME/lib $MQM_HOME/lib/libmqmcbt.o -lmqm"
```

---

## Using CICS with WebSphere MQ

CICS supports access to WebSphere MQ queues by using MQ API calls in CICS programs through XA or non-XA connections. For information about this, see the *CICS Application Programming Guide* and *WebSphere MQ Application Programming Guide*.

The following procedure describes how to enable the CICS region to connect to a WebSphere MQ queue manager. It describes XA connection, non-XA connection, Micro Focus Server Express COBOL runtime requirements, and ACUCOBOL-GT runtime requirements. (For information about using Micro Focus Server Express COBOL with CICS, see Chapter 12, "Preparing CICS for use with Micro Focus Server Express COBOL," on page 175. For information about using ACUCOBOL-GT with CICS, see Chapter 13, "Preparing CICS for use with ACUCOBOL-GT," on page 177.)

### Prerequisite tasks and conditions

The WebSphere MQ queue manager must be started before the region is started. Otherwise, the XA\_OPEN command fails when the region attempts to connect.

### Procedure for setting up the CICS interface to WebSphere MQ

This section describes how to enable transactions that are to be executed from within CICS in an XA environment. You can enable the use of Micro Focus Server Express COBOL or ACUCOBOL-GT by including the steps that are associated with the particular COBOL product.

**Step 1.** Prepare the WebSphere MQ product.

- a. Enable connections between CICS and the WebSphere MQ product.  
No additional product configuration is required.
- b. Grant privileges that are required for use of the WebSphere MQ product with CICS.

User **cics** must be in the **mqm** group.

If access to the queue or to any object in the queue is restricted, user **cics** must be given the appropriate MQI authorizations as required by the transaction. For information about this, see the *WebSphere MQ Security* and *WebSphere MQ System Administration Guide*.

**Step 2.** For XA connections, build the WebSphere MQ switch load file.

Source file \$MQM\_HOME/samp/amqzscix.c is included with WebSphere MQ to enable you to build your own switch load file. You must install "WebSphere MQ Samples" component to have access to the switch load source file.

Use the following command to rebuild the switch load file that is using the source file (where \$MQM\_HOME is the installation directory for WebSphere MQ):

- **On AIX:**

```
xlc_r amqzscix.c -I/usr/include -I/usr/lpp/cics/include \
-e amqzscix -o amqzsc /usr/lpp/cics/lib/regxa_swa.o \
-L$MQM_HOME/lib -L/usr/lpp/cics/lib -lmqm_r -lmqmcics_r \
-lmqmxa_r -lcicsrt -lEncServer -lpthreads -lcicsipc
```

- **On HP-UX:**

```
cc -c -s amqzscix.c -Aa +z -o amqzscix.o ld -b -o amqzsc amqzscix.o \
/opt/cics/lib/regxa_swa.o +e CICS_XA_Init \
-L$MQM_HOME/lib -L/opt/cics/lib -lmqmxa_r -lmqm_r \
-lEncServer -lcicsipc -lpthreads -ldbm -lm -lc
```

- **On Solaris:**

```
cc -s amqzscix.c -G -o amqzscix -eCICS_XA_Init -L/opt/cics/lib \
-L$MQM_HOME/lib /opt/cics/lib/regxa_swa.o -lmqmcics -lmqmxa \
-lmqm -lmqmcs_d -lmqmzse -lcicsrt -lEncServer -lcicsipc -lpthreads
```

Step 3. If you are going to use Micro Focus Server Express COBOL, go to step 4. If you are going to use ACUCOBOL-GT, go to step 6. Otherwise, go to step 7.

Step 4. Build the Micro Focus Server Express COBOL runtime.

To make the COBOL runtime unique to the CICS region, build the runtime in the region's bin directory. If COBOL applications that are on this region also contain SQL statements, you can add the database libraries to the **cicsmkcobol** command shown below.

For CICS/MQ COBOL applications, the COBOL runtime is created as shown (where \$MQM\_HOME is the installation directory for WebSphere MQ):

```
cicsmkcobol -L$MQM_HOME/lib $MQM_HOME/lib/libmqmcbrt.o -lmqm_r
```

Step 5. Go to step 7.

Step 6. Link the WebSphere MQ libraries with ACUCOBOL-GT.

You can link the WebSphere MQ libraries by using the ACUCOBOL-GT configuration variable, **SHARED\_LIBRARY\_LIST** (which can also contain database libraries that you are going to use). You do not need to relink the ACUCOBOL-GT runtime or build the language method file unless you are adding customizations that are not distributed as shared libraries, or you have modified the .c files in \$ACUCOBOL/lib. Refer to the ACUCOBOL-GT documentation for more information.

Step 7. Prepare the region for WebSphere MQ.

a. Set up the region's environment.

The environment is passed to the region in one of two ways:

- From the shell in which the region is started.

- **On AIX**

If you are *not* using ACUCOBOL-GT, verify that root's LIBPATH includes the following library search paths:

MQM\_HOME/lib:COBDIR/lib:/usr/lib. The LIBPATH environment variable should be set in the /etc/environment file.

If you are using ACUCOBOL-GT, verify that root's LIBPATH includes the following library search paths:  
MQM\_HOME/lib:\$ACUCOBOL/lib:/usr/lib. The LIBPATH environment variable should be set in the /etc/environment file.

- **On HP-UX**

Verify that root's SHLIB\_PATH includes the following library search paths: MQM\_HOME/lib:COBDIR/lib:/usr/lib.

- **On Solaris**

Verify that root's LD\_LIBRARY\_PATH includes the following library search paths: MQM\_HOME/lib:COBDIR/lib:/usr/lib.

- From the region's environment file.

- **On AIX**

LIBPATH=MQM\_HOME/lib:COBDIR/lib:/usr/lib

- **On HP-UX**

SHLIB\_PATH=MQM\_HOME/lib:COBDIR/lib:/usr/lib

- **On Solaris**

LD\_LIBRARY\_PATH=MQM\_HOME/lib:COBDIR/lib:/usr/lib

(Where MQM\_HOME is the installation directory for WebSphere MQ, and COBDIR is the installation directory for Micro Focus Server Express COBOL.)

- b. For XA connections, add the Product Definitions (XAD) to the region.

Create an XAD for the region as shown:

```
cicsadd -c xad -r regionName -P "MQMXA"
ResourceDescription="MQM XA Product Definition"
XAOpen="queueManagerName"
XASerialize="multiple_association"
SwitchLoadFile="MQM_HOME/lib/amqzsc"
```

where:

- MQM\_HOME is the installation directory for WebSphere MQ
- *regionName* is the name of the region
- *queueManagerName* is the name of the WebSphere MQ queue manager (this must match the value used with the **strmqm queueManagerName** command)

Because WebSphere MQ/XA is thread-aware, the XA connections should be established with multiple\_association mode. To do this you must set XASerialize="multiple\_association" in the XAD definition, as shown above.

- c. Install the "Task termination exit sample" that is provided by WebSphere MQ. For information about the "Task termination exit sample" and its installation procedures, see the *WebSphere MQ System Administration Guide*.

**Step 8.** For XA connections, verify the connection.

CICS logs XA messages in the region's console file. To determine whether the region can connect to the WebSphere MQ queue manager:

- a. Start your CICS region. If this is a newly created region, follow the steps that are given in Chapter 4, "Working with a file manager in a CICS region," on page 23; otherwise, enter the command:

```
cicscp start region regionName StartType=cold
```

- b. In a second window, use this command to view the console messages:

```
cicstail -r regionName
```

If XA problems occur, check for additional information in console.nnnnnnn or symrecs.nnnnnnn logs that are under the region's directory.

**Step 9. Test the connection.**

You can use the WebSphere MQ supplied example programs, amqscic0.ccs and amqscic0.tst to test the connection between CICS and WebSphere MQ. To access the sample programs, install the "WebSphere MQ Samples" product component. For more information about how to run the samples see the *WebSphere MQ Application Programming Guide*.

## Signal handling when working with WebSphere MQ

TXSeries for Multiplatforms is a transaction processing monitor. If TXSeries for Multiplatforms is configured with WebSphere MQ as the resource manager, by default the signals raised from the WebSphere MQ modules are caught by the WebSphere MQ signal handler. Under the TXSeries for Multiplatforms environment, because CICS is a Transaction Processing Monitor, CICS coordinates the resources, hence the signal must be caught by the TXSeries for Multiplatforms signal handler. If WebSphere MQ is running connected to CICS the WebSphere MQ, you must disable the signal handler. For WebSphere MQ Versions 5.2 and 5.3 the environment variable used to disable the WebSphere MQ signal handler is MQS\_NO\_SYNC\_SIGNAL\_HANDLING=1, which is set in the /etc/environment file.





---

## Chapter 15. Customizing CICS with user exits and user programs

This chapter describes how to use user exits and user programs to configure CICS. Most of the user programs that are described in this chapter are CICS-supplied programs, which can be modified for your needs. Post-initialization and shutdown programs are not CICS-supplied programs.

The *TXSeries for Multiplatforms Administration Reference* contains the reference information for user exits and the CICS-supplied user programs.

“Working with CICS user exits” introduces the CICS-supplied user exits, (task termination, dump request, dynamic transaction routing, distributed program links, and syncpoints), and it explains how to use these user exits.

“The user programs” on page 188 explores the CICS-provided user programs: External Security Manager, terminal autoinstall, CICS client autoinstaller, performance monitoring, post-initialization user programs, and shutdown user programs.

---

### Working with CICS user exits

A *user exit* (also referred to as a *user exit point*) is the point in a CICS program at which CICS can transfer control to a program that you have written (a *user exit program*), and can then resume control when your program has finished its work. You do not have to use any of the user exits, but they are useful if you want to extend and customize the function of your CICS system to meet your specific requirements. The *TXSeries for Multiplatforms Intercommunication Guide* and the *TXSeries for Multiplatforms Administration Reference* contain detailed information about user exits.

CICS on Open Systems supplies the following user exits:

- Task termination

The task termination user exit can be invoked at normal and abnormal task termination (after any syncpoint has been taken).

- Dump request

The dump request user exit can be invoked at the following points:

- When the EXEC CICS DUMP command is issued
- When a transaction abend occurs
- When a CEMT PERF SNAP is requested
- When the region is shut down
- When a system abend occurs

- Dynamic transaction routing

The dynamic transaction routing user exit can be invoked when a request is made to route a transaction. This user exit is used to route transactions dynamically, based on conditions that are specified in the user exit program.

- Dynamic distributed program link

The dynamic distributed program link user exit can be invoked when a request is made to link to another program. This user exit is used to link a program dynamically, based on conditions that are specified in the user exit program.

- Syncpoint  
The syncpoint user exit can be invoked when a transaction issues an EXEC CICS SYNCPOINT.

## Implementing user exits

Before you attempt to make use of a user exit, read the following information:

- “Associating an exit program with its user exit.”
- “Rules and guidance for writing user exit programs” on page 187.

### Associating an exit program with its user exit

A semistatic method of associating exit programs with user exits is available. It is based on the **UserExitNumber** attribute in the Program Definitions (PD). These numbers are shown in Table 31. This method provides the following:

- A generic method of associating a single program with a user exit
- A generic method for invoking user exits
- A standard naming convention for user exits
- A structure for the standard parameters passed when a user exit is invoked

**Note:** Although the equivalent CICS TS for Windows exits are identifiable by the standard naming convention, consider the behavior of the user exit and the parameter list that are passed as similar rather than identical, because of differences in the underlying CICS environment.

The association of a program with a user exit is held as a table in the region control area (RCA) and is visible in system dumps. A user exit is regarded as active only if it has a program associated with it in the RCA table. This relationship is passed to new application servers as they are created, but not to existing application servers. Where a program is already associated with the user exit, this request supersedes any previous request for new application servers.

The user exit-program relationship is removed for new application servers when the PD is uninstalled. The user exit is regarded as inactive when no PD exists for the user exit program; this applies to *all* application servers. If the user exit is later reactivated for a different PD, this new user exit-program association applies only to new application servers. Those application servers that already exist continue to use the previous user exit-program association. It is *not* possible to copy a new user exit program for an existing application server.

A local copy of the table is in the task control area (TCA), and it is visible in system and transaction dumps. This copy is created when a new application server is initialized, and its contents remain “static” until the application server is terminated. The user exit programs are loaded at this time to save the performance overhead of doing this each time the user exit is called. If a problem occurs during the loading of the user exit program, the error message ERZ015043W is displayed, and the user exit is disabled for that application server only; no later attempt is made to load the program.

**Note:** To ensure that the same program-user exit relationship is actively used in all application servers, you are advised to shut down and restart CICS whenever an existing user exit program is to be replaced by another program or a new version of the same program.

Table 31. User exit numbers

| User exit number | Description         | User exit name |
|------------------|---------------------|----------------|
| 13               | Program autoinstall | UE015013       |

Table 31. User exit numbers (continued)

| User exit number | Description                      | User exit name |
|------------------|----------------------------------|----------------|
| 15               | Task termination                 | UE014015       |
| 17               | Dump request                     | UE052017       |
| 25               | Dynamic transaction routing      | UE014025       |
| 33               | Dynamic resource definition      | UE046033       |
| 50               | Dynamic distributed program link | UE015050       |
| 51               | Syncpoint                        | UE016051       |

Table 31 on page 186 lists the description and name for each user exit number:

- User exit number: The number that is used in the PD **UserExitNumber** attribute.
- Description: A description of the user exit.
- User exit name: Each user exit has a unique name and is at a point in the module at which it can be useful to do some extra processing. The format of a user exit name is UEcccnnn, where *ccc* is the number of the CICS component that invokes the user exit program and *nnn* is the user exit number, padded to three characters with leading zeros.

## Rules and guidance for writing user exit programs

**On the AIX platform.** A sample user exit program is provided in the `/usr/lpp/cics/src/samples/userexit` directory.

**On the HP-UX or Solaris platforms.** A sample user exit program is provided in the `/opt/cics/src/samples/userexit` directory.

Consider the following points when writing user exit programs:

- A user exit program must be written in C.
- A user exit program must be reentrant.
- Threads must not be switched in the user exit program.
- Use of mutexes for shared resources can cause lockouts.

**Note:** A *mutex* is a *mutual exclusion mechanism* that is intended for short-term use with internal data structures (as opposed to recoverable data) within an application.

- It is the user's responsibility not to damage the integrity of the CICS system.
- A user exit program must not use EXEC CICS commands (the Application Programming Interface).
- The supplied function prototype for a particular user exit must be included in the user exit program.
- The supplied exit-specific structure for a particular user exit must be included in the user exit program.
- The supplied `cics_UE_Header_t` structure must be used in all user exit programs.
- All documented restrictions on writing CICS application programs (see the *TXSeries for Multiplatforms Application Programming Guide*) also apply to writing user exit programs. Also, do not use system calls.
- Do not use the user exit program as a CICS application program because the results can be unpredictable.
- No specific facilities are provided for handling errors in user exit programs.

- The user exit program must set a return code that is defined as being supported for the user exit; otherwise, the user exit program takes the default action, which in many cases causes the user exit to be disabled for the Application Server.
- No facilities are provided for accessing CICS resources within a user exit program.
- Where a retry mechanism is provided for a user exit, this is clearly stated as part of the user exit definition and a parameter is provided to indicate that the user exit is being redriven.
- Version numbers are provided on all structures that are provided for use in user exit programs. You must ensure that the user exit program checks the version number and uses the correct version of the structure.
- A task work area is provided for sharing between user exits within an application server that use the same TCA. You must ensure no conflicts exist in the usage of fields within this task work area. A pointer to the task work area is passed to all user exits. The work area is initialized between tasks.
- The length of the task work area is fixed at 128 bytes and cannot be changed.
- Signal handlers must not be included in user exit programs.
- The use of a user exit program to interface with other software packages can be unpredictable and unsupported. Check any documented requirements before implementing a user exit of this type in a CICS production environment.
- The standard header structure (**cics\_UE\_Header\_t**), constants, data definitions, exit-specific structures, and the function prototypes for the user exits are all found together in the include file **cicsue.h**. The **cicsue.h** file must be included in all user exit programs.
- All user exits are called with shared memory attached. You must ensure that the user exit code does not compromise the integrity of the CICS system.
- The entry point for all user exit programs is **cics\_UE\_entry**.
- All user exits are called with two parameters:
  - Pointer to the **cics\_UE\_Header** parameter list
  - Pointer to the exit-specific parameter list for the user exit that has been invoked.
- Where an enum parameter in a user exit parameter list is defined as a typedef in **cicsue.h**, user exit programs must use the supplied data names when referencing the parameter to avoid unpredictable results.
- References are made to the results of setting a parameter to a null string. A null string is a string that begins with a null character `'/0'`. For example, a null string can be placed into the system ID parameter of the parameter list of user exit UE014025 as follows:
 

```
strcpy (UE_specificptr->UE_Dp1sysid, "");
```

All other strings that are returned by the user exit must also be null terminated.

### Migration and coexistence

To assist migration from other CICS family members, the user exit naming convention incorporates the equivalent CICS OS/2 or CICS for Windows exit number, where that exists.

---

## The user programs

User programs make it possible for you to modify the system without changing any of its standard user interfaces. These programs can be customized or replaced to suit the requirements of your installation.

## Using the External Authentication Manager user program

The default External Authentication Manager (EAM) program, **cics\_eam**, is supplied as part of CICS. You can implement it in its unmodified form, or you can customize it to meet the particular requirements of your application.

The initial configuration of a CICS region sets the **EAMLoad** attribute in the Region Definitions (RD) to **no**. This action specifies that internal authentication is performed. If you want to override internal authentication, you must modify the EAM program to perform the required user authentication checks. After you have modified the program, you must place the generated executable load module into the required load library, so that it is available for use by the CICS system. You must then set the EAM attributes in the RD to make your program available to the region.

For more information, see “Using an External Authentication Manager” on page 91, the *CICS Administration Guide*, and the *TXSeries for Multiplatforms Application Programming Guide*.

## Using the External Security Manager user program

The default External Security Manager (ESM) program, **cics\_esm**, is supplied as part of CICS. You can implement it in its unmodified form, or you can customize it to meet the particular requirements of your site. Note that in its unmodified form, the ESM simply returns **CICS\_ESM\_RETURN\_OK** to every security check, indicating that authorized access has been granted to the transaction or resource.

The initial configuration of a CICS region sets the **ESMLoad** attribute in the Region Definitions (RD) to **no**, therefore specifying that internal security checking is performed. If you want to override internal security checking, you must modify the ESM program to perform the security checks that you require.

After modification, you must place the generated executable load module into the required load library, so that it is available for use by the CICS system. You must then set the ESM attributes in the RD to make your program available to the region.

For more information, see “Using an External Security Manager” on page 106, the *TXSeries for Multiplatforms Administration Reference*, and the *TXSeries for Multiplatforms Application Programming Guide*.

## Using the terminal autoinstall user program

The Terminal Definitions (TD) are used to define terminals to CICS. Only a few terminal entries are stored permanently in the WD. When these terminals request a CICS session, CICS must create a WD entry automatically so that CICS can recognize the terminal. CICS does this by selecting a model name from a list of models. Each model has an entry in the WD. This entry is used to create a runtime WD entry for the terminal.

These autoinstalled terminals are defined in the runtime only while they are connected to CICS. When you terminate the connection, CICS automatically removes the WD entry. A particular advantage of autoinstall is that you do not have to create and maintain a large permanent WD. Any terminal or pseudoterminal that does not have an entry in the WD for the region can be autoinstalled.

CICS can use the terminal autoinstall user program to control the process of adding entries to the runtime WD whenever terminals that are not already defined to CICS request connections to CICS.

The supplied user program selects the first in a list of suitable models to derive the terminal identifier. You can customize the program to carry out any processing that you require. For example, you can count and limit the total number of terminals that are logged on, or you can keep utilization information about the terminal.

The user program is also called when the terminal is uninstalled and removed from the WD. This can be used, for example, to update any statistics set up on autoinstall.

The terminal autoinstall user program is not called when a shipped terminal is installed. A shipped terminal is one that has a terminal definition associated with it when it is sent from the local CICS system. The **IsShippable** attribute in the WD stanza must be set to **yes** for the terminal definition to be shipped with the install request from the local system.

See the *TXSeries for Multiplatforms Administration Reference* for more information.

## Using the performance monitoring user program

Specify whether you require performance monitoring, by using the CICS resource management commands to set the **MonitorStatus** attribute in the permanent Monitoring Definitions (MD). You can also use the CEMT SET MONITOR transaction to set the **MonitorStatus** attribute. You can use **cicsupdate** to specify the name of the Performance Monitoring user program in the **UserMonitorModule** attribute in the MD.

**On the AIX platform.** You can also use System Management Interface Tool (SMIT) to specify the Performance Monitoring user program.

CICS calls the program under the following conditions:

- During task startup
- During task exit
- To process the EXEC CICS ENTER command
- To stop the user clocks and return the user data that is to be written to the monitoring transient data queue
- To restart the user clocks
- At the end of a task, to clean up the user data

**Note:** If a CICS application server that is in an XA environment terminates with an abend A147, a monitor record is not written. If a transaction is purged with an abend A147, a monitor record is written.

You can customize the default program to perform any processing that you require.

See “Monitoring performance” on page 219 and the *TXSeries for Multiplatforms Administration Reference* for more information.

## Using post-initialization user programs

Post-initialization programs are user-written programs that are specified in the Region Definitions (RD). CICS runs these programs sequentially and automatically,



after completing the initialization phase but before allowing users to sign on. You must also specify an entry in the Program Definitions (PD) for each post-initialization program.

Post-initialization programs that are user written allow you to implement functions that are specific to your system, and show your unique requirements. For example, you can issue an EXEC CICS START command from within a post-initialization program to perform a long-running background task. When writing these programs, however, you must remember that they run before CICS is generally available to users. This imposes particular limitations on their functions.

### Invoking user-written programs at startup

For post-initialization programs to be identified to CICS, they must have a PD entry defined for them. Use the RD **StartupProgList** attribute to define the sequence in which they are to run. These definitions are described in the *TXSeries for Multiplatforms Administration Reference*.

CICS runs each post-initialization program sequentially at the completion of CICS initialization and before CICS allows any user to sign on.

You can run up to 10 programs in this way. If this limit is too small for your processing requirements, you can run more by using the EXEC CICS LINK and EXEC CICS XCTL commands within each program.

### Guidelines for writing post-initialization programs

When writing programs for CICS to run during the post-initialization phase, consider the following factors:

- All CICS facilities, except direct communications with terminals, are available to post-initialization programs. Because intersystem communication has pseudoterminal entries associated with its function, you cannot run any intersystem communication (ISC) function during the processing of a post-initialization program at the time a region is initialized.
- You must define all user-written programs for CICS to run during post-initialization in the Region Definitions (RD) and in the Program Definitions (PD). Programs that CICS initiates from the RD list have no access to a Transaction Work Area (TWA).
- Post-initialization programs must not request any service that can logically suspend or abnormally terminate the task in which they are running.
- Post-initialization programs must not depend on transactions being initiated by interval control, because no interval-control initiations occur until CICS completes the post-initialization phase.
- Post-initialization programs must not explicitly issue an abnormal termination. Also, you must be aware of implied abnormal terminations that can be issued on your behalf if you do not code an EXEC CICS HANDLE CONDITION.
- Post-initialization programs must not issue dump requests.
- Although you can issue EXEC CICS START commands during post-initialization processing, CICS queues the tasks, and these tasks do not start running until CICS has completed post-initialization processing.

## Using shutdown user programs

You specify in the Region Definitions (RD) which user-written programs CICS is to run during shutdown. CICS runs these programs sequentially during the first and second quiesce stages of normal shutdown processing. CICS starts these programs

automatically during the shutdown process. You must also create an entry in the Program Definitions (PD) for each of the shutdown programs.

### Invoking user-written programs at shutdown

CICS allows two lists of user-written programs to run automatically at different stages of normal shutdown processing. You must define both lists in the Region Definitions (RD), in addition to defining each shutdown program in the Program Definitions (PD).

During the first quiesce stage of normal shutdown processing, CICS automatically runs, in sequence, the first list of user-written shutdown programs, as specified in the **ShutdownProgList1** attribute in the RD. During the second quiesce stage, CICS automatically runs the second list of user-written shutdown programs, as specified in the **ShutdownProgList2** attribute in the RD.

You can run up to 10 programs in each quiesce stage in this way. If this limit is too small for your processing requirements, you can run more by using the EXEC CICS LINK and EXEC CICS XCTL commands within each program.

### Guidelines for writing shutdown programs

When writing programs for CICS to run during normal shutdown, remember that these programs run either in the first or second quiesce stages of normal shutdown only.

- First quiesce stage: During the first quiesce stage, terminals are still available, but CICS rejects transactions that are started by terminal input unless the transactions are flagged in the Transaction Definitions (TD). CICS allows transactions that are started by other ways, for example, Automatic Transaction Initiation (ATI), to proceed.

The first quiesce stage is complete when CICS has run the first-stage shutdown programs that are listed in the Region Definitions (RD), and when no user tasks are running.

- Second quiesce stage: The second quiesce stage does not begin until all region activity in the first quiesce stage is complete. Shutdown then continues, and no new transactions are allowed to start. When all programs that are defined to be run in the second quiesce stage have completed, CICS stops.

#### Notes:

1. Programs that are specified for the second quiesce stage cannot use any terminal control services or start new transactions. CICS does not allow such requests and writes a message to the system log. CICS does not allow any automatically initiated transactions to start during the second quiesce stage.
2. If your user shutdown program abnormally terminates, CICS performs a syncpoint rollback.
3. Ensure that your user shutdown programs handle all abnormal termination conditions.

For information about how to restore CICS after an abnormal shutdown, see Chapter 20, “Understanding CICS backup and recovery,” on page 239.

### How to start new transactions at shutdown

If you need to start new transactions during CICS shutdown, you can do this only during the first quiesce stage of the normal shutdown process. The purpose of starting new transactions at shutdown depends upon the type of user and your particular system, but examples include starting transactions to collect statistics,

signing off from the CICS region, and using CEMT to purge long-running tasks that are delaying the completion of the first quiesce stage of shutdown.

To start a transaction at shutdown:

- CICS must be in the first quiesce stage of normal shutdown.
- You must set the **InvocationMode** attribute in the Transaction Definitions (TD) entry to **at\_shutdown\_start** to define the transaction as capable of being started during shutdown, or the transaction must be automatically initiated.

Start the transaction from a terminal as normal.



---

## Part 3. Maintaining your CICS region

This part of the guide discusses the following operational tasks with which a system administrator is continually involved:

- Chapter 16, "Starting up a CICS system," on page 197
- Chapter 17, "Shutting down a CICS system," on page 207
- Chapter 18, "Monitoring your CICS system," on page 213
- Chapter 19, "Improving the performance of your CICS system," on page 223
- Chapter 20, "Understanding CICS backup and recovery," on page 239



---

## Chapter 16. Starting up a CICS system

This chapter describes how a CICS system is started.

“The startup sequence” describes the standard startup procedures.

“Cold starts and warm starts” on page 198 describes what happens during the types of starts that you can use on your CICS system.

“Using the CICS control program commands for startup” on page 199 describes the use of the **cicscp** command scripts to start a CICS system.

“Startup procedures” on page 199 describes the startup of the Structured File Server (SFS) server and the CICS region.

“Recovery and restart for CICS startup” on page 200 describes how CICS handles recovery and automatic restarts after various types of shutdowns.

“File manager unavailable at startup and while running transactions” on page 201 lists the problems that can occur if the file manager is unavailable at startup.

---

### The startup sequence

A CICS system is started in this sequence:

1. File Manager: The file manager must be started before any region that uses it is started. See “File manager unavailable at startup and while running transactions” on page 201.
2. Local SNA: If a region is using local Systems Network Architecture (SNA) support, SNA must be started before the region is started. The following list shows where you can get information about using a communications product for your platform:
  - **On the AIX platform.** See *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS* and the administration documentation for Communications Server for AIX.
  - **On the HP-UX platform.** See *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS* and the administration documentation for HP-UX SNAplus2.
  - **On the Solaris platform.** See *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS* and the administration documentation for SNAP-IX.
3. PPC Gateway server: If any region uses a PPC Gateway server, the PPC Gateway server must be started before the regions that use it are started. This assumes that SNA is also started. See the *TXSeries for Multiplatforms Intercommunication Guide* if the PPC Gateway server is running on a machine that also has CICS installed on it.
4. Regions: The regions can be started when all the required file manager, PPC Gateway server, and SNA components are started.

When the region and its components are started, connect to the region by starting a client, as described in Chapter 6, “Working with CICS clients,” on page 71. Enter the CICS-supplied transaction CEMT to activate the CICS master terminal display.



The remainder of this chapter describes how to use the CICS control program (**cicscp**) to start up CICS. For a description of the full functions of the **cicscp** commands, see the *TXSeries for Multiplatforms Administration Reference*.

Alternatively, you can use the commands that underlie the **cicscp** commands (see Appendix B, “Using CICS resource management commands for system management,” on page 269).

**On the AIX platform.** You also can use the System Management Interface Tool (SMIT) to start your system (see Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259).

---

## Cold starts and warm starts

When you start a CICS region automatically, CICS starts up in one of the following ways, depending upon how the region terminated previously:

- **Cold start:** The region has not previously been started. When you specify that a CICS region is to be *cold started*, no attempt is made to recover any previously running transactions, and any data that is stored in the transient data queues and temporary storage queues is lost. That is, the region starts as if for the first time.
- **Warm start:** The previous run of the CICS region terminated normally. CICS retains any data that is stored in the transient data queues and temporary storage queues.
- **Emergency start:** The previous run of the CICS region terminated abnormally, for example, with an immediate shutdown or a hardware error. CICS performs any necessary recovery processing and retains any data that is stored in the recoverable transient data queues and temporary storage queues, but discards data that is in the nonrecoverable queues.

The first time a region is started after its creation, it must be cold started to ensure correct initialization.

You can start CICS in either the default region or in a specific region by specifying a region name as part of the startup parameters.

You can start only one CICS region at a time. You can start as many CICS regions as you like; however, the performance of the CICS regions depends on the overall system load.

If you want to run a region with a specific locale, you can set up the locale in the region’s environment file, which is in `/var/cics_regions/regionName/environment`. See “Setting the language for the region” on page 21.

After CICS initialization is complete, before CICS allows users to sign onto the region, CICS automatically runs any user-written post-initialization programs.

## Cold starting the SFS server

Cold starting the SFS server discards all files on the SFS logical volumes. The data that is in the files, and the files themselves are lost. Each time that the SFS server is cold started, these files must be re-created before the server can be used. These include all files that contain transient data queues, temporary storage queues, unexpired START requests, and application data files (files defined by the File Definitions (FD)). Therefore, cold starts are not recommended for the SFS server.

If you must cold start the SFS server, see “Administration of file and queue management on an SFS server” on page 23 for information about re-creating these files. Related topics include “Using post-initialization user programs” on page 190 and “Authorizing access to startup and shutdown programs” on page 99.

---

## Using the CICS control program commands for startup

The **cicscp** commands performs as many of the following actions as necessary:

- Sets up IBM CICS Clients.
- Creates a user ID for the SFS server (AIX only).
- Creates logical volumes for the SFS server (AIX only).
- Starts the file manager if it is an SFS server.
- Configures the file manager for the region.
- Configures the CICS region.
- Starts the CICS region.

It is recommended that inexperienced users of **cicscp** commands use the **-v** and **-l** options, to understand and monitor what the command does. The **-v** option causes the command to show the output of the commands that are used. The **-l logfile** option causes the command to produce a log of activity in *logfile*. For example, the following command starts a region named **regionA** with a log file named **regionA.log**.

```
cicscp -v -l regionA.log start region regionA
```

If you need a nondefault configuration, you can explicitly create it by specifying options with the **cicscp** command that supply the necessary overrides to the default values.

---

## Startup procedures

This section describes how to start up a CICS system by using the **cicscp** command. For alternative procedures, see Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 or Appendix B, “Using CICS resource management commands for system management,” on page 269.

### Starting an SFS server by using the **cicscp start sfs\_server** command

The following procedure starts up an SFS server by using the **cicscp -v start sfs\_server** command.

1. Log with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Enter the command:

```
cicscp -v start sfs_server SFS_server_name
```

where *SFS\_server\_name* is the name of the SFS server.

### Starting the region by using the **cicscp start region** command

The region relies on the underlying components for operation. Therefore, you must start up the file manager before the region is started.

**Note:** You can keep track of the region startup sequence by opening another window and issuing the command:

```
tail -f /var/cics_regions/regionName/console.
```

1. Log with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use this command to start up the region:

```
cicscp -v start region regionName
```

where *regionName* is the name of the region.

## Starting CICS by using the cicscp start all command

This procedure uses the **cicscp start all** command to start up all CICS objects that are configured on the local host.

1. Log with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Enter the command:

```
cicscp -v start all
```

---

## Recovery and restart for CICS startup

In a cold start, the startup of the region occurs without regard to any previous region activity. For an automatic start, CICS starts the region depending on the state at the last shutdown. The following can result:

- A cold start, if you have not started the region before
- A warm start (autostart), if the region terminated with a normal shutdown
- An emergency start, if the region terminated with an immediate shutdown or due to an abnormal termination

## Recovering resource definitions at startup

If you restart CICS with an autostart, CICS recovers the latest resource definitions. This means that CICS restarts with exactly the same resources regardless of the method that was used to terminate the previous CICS run. If you want to restart CICS with your permanent set of resource definitions, you must cold start CICS. If you start CICS with your permanent set of resource definitions, you cannot revert to any previous definitions.

**Note:** CICS does not restore any changes that are made through the runtime resource management transaction, CEMT, or the EXEC CICS SET commands.

## Restarting CICS

If a region terminates before all transactions have completed normally, when you next start the region, CICS must restore the recoverable resources of the region to a known consistent state before normal processing can resume.

- If a transaction did not finish preparing for a synchronization point, CICS discards the changes that were made (backs them out).
- If a transaction prepared for a synchronization point but did not commit the synchronization point at the time the region terminated, CICS prepares the changes and waits for a Logical Unit of Work (LUW) outcome.

- If a transaction completed a synchronization point, the changed resources are in a consistent state, and no further action is required.

If a transaction prepares successfully but is not committed, CICS saves, in the system log file, the changes that were made by that transaction, to allow the changes to be reapplied if the region terminates abnormally. During recovery, CICS reads the system log and again prepares the transaction. If everything performs normally, CICS commits the changes and leaves the data in a consistent state.

## Recovery control processing

CICS collects the following information:

- Information relating to in-flight logical units of work and tasks
- Information relating to completed LUWs and tasks

## Backout processing

CICS and the file manager cooperate to back out the effect of in-flight transactions on the following resources:

- Recoverable transient data queues
- Recoverable temporary storage queues
- Recoverable data files
- EXEC CICS START PROTECT requests

See Chapter 20, “Understanding CICS backup and recovery,” on page 239 for more information.

---

## File manager unavailable at startup and while running transactions

This section contains the following tables:

- “Effect of an unavailable file manager at CICS startup”
- “Effect of an unavailable file manager at various stages” on page 204
- “Effect of other unavailable file managers” on page 205

## Effect of an unavailable file manager at CICS startup

Table 32 shows the effect of an unavailable file manager at startup:

*Table 32. Effect of an unavailable file manager at CICS startup*

| File manager holding .... is unavailable | CICS performing a cold start                                                                                                                                                                                                                                | CICS performing an automatic start |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| One or more data files.                  | Files that CICS would have opened are instead marked CLOSED, UNENABLED. Message ERZ021004E is logged for each such file. Each file can be opened when the file manager becomes available, through explicit use of SET FILE or CEMT. CICS continues startup. | As for cold start.                 |

Table 32. Effect of an unavailable file manager at CICS startup (continued)

| File manager holding .... is unavailable                                           | CICS performing a cold start                                                                                                                                                                                        | CICS performing an automatic start                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Protected locally queued transactions [START PROTECT NOCHECKs are held on a file]. | CICS needs to clear this file. Because it cannot do so, it sets an indicator for this to be done through an automatically scheduled CICS-private transaction. Message ERZ061013E is logged. CICS continues startup. | CICS schedules two CICS-private transactions: one is to send the locally queued requests to their remote systems, and the other is to purge aged requests. Message ERZ061013E is logged. These transactions also log this message until the file becomes available. CICS continues startup.                                                                                                  |
| Nonprotected locally queued transactions [START NOCHECKs are held on a file].      | As for protected locally queued transactions.                                                                                                                                                                       | As for protected locally queued transactions.                                                                                                                                                                                                                                                                                                                                                |
| Logically recoverable transient data queues.                                       | CICS needs to empty the file. Message ERZ022132W is logged. The file is opened later on the first access after the file manager becomes available, when all previous entries are deleted. CICS continues startup.   | If the automatic start follows normal shutdown, CICS recovers its queue pointers from LOG records, without needing the file. No message is logged. The file is opened later on the first access after the file manager becomes available. CICS continues startup. Otherwise, CICS needs to recover its queue pointers from the file. Message ERZ022124E is logged and startup is terminated. |
| Physically recoverable transient data queues.                                      | As for logically recoverable transient data queues.                                                                                                                                                                 | As for logically recoverable transient data queues.                                                                                                                                                                                                                                                                                                                                          |
| Nonrecoverable transient data queues.                                              | As for logically recoverable transient data queues, except message ERZ022123E is logged.                                                                                                                            | If the automatic start follows a normal shutdown, as for logically recoverable transient data queues; otherwise, as for cold start.                                                                                                                                                                                                                                                          |
| Recoverable temporary storage queues that are owned by users.                      | CICS needs to empty the file. Message ERZ023195W is logged. The file is opened later on the first access after the file manager becomes available, when all previous entries are deleted. CICS continues startup.   | If the automatic start follows a normal shutdown, CICS recovers its queue pointers from log records, without needing the file. No message is logged. The file is opened later on the first access after the file manager becomes available. CICS continues startup. Otherwise, as for logically recoverable transient data queues, but message ERZ022372E is logged.                         |

Table 32. Effect of an unavailable file manager at CICS startup (continued)

| File manager holding .... is unavailable                                  | CICS performing a cold start                                         | CICS performing an automatic start                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nonrecoverable temporary storage queues that are owned by users.          | As for recoverable temporary storage queues that are owned by users. | If the automatic start follows a normal shutdown, as for recoverable temporary storage queues that are owned by users; otherwise, as for cold start.                                                                                                                                                                                                                                                                                                                                                                                                              |
| Recoverable temporary storage queues that are owned by timer services.    | As for recoverable temporary storage queues that are owned by users. | <p>CICS needs to rebuild its ICQ from records that are held in the file. If the automatic start follows a normal shutdown, the file cannot be opened and CICS defers this work until later. If the file can be opened, it is possible that one or more records cannot be read because they are waiting for transaction resolution. CICS notes such records and rebuilds ICQ entries from them later. CICS continues startup.</p> <p>If the automatic start follows an abnormal shutdown, as for recoverable temporary storage queues that are owned by users.</p> |
| Nonrecoverable temporary storage queues that are owned by timer services. | As for recoverable temporary storage queues that are owned by users. | <p>If the automatic start follows a normal shutdown, CICS needs to restore the ICQ for nonprotected START requests from this file; otherwise, as for cold start.</p> <p>If the file manager is not available, message ERZ023103E is logged and startup is terminated. If the file is available but cannot be read, message ERZ023104E is logged and startup continues.</p>                                                                                                                                                                                        |
| Recoverable temporary storage queues that are owned by terminal services. | As for recoverable temporary storage queues that are owned by users. | If the automatic start follows an abnormal shutdown, CICS must possibly need to restore some autoinstalled terminals. Definitions for these reside in recoverable temporary storage queues that are held in the file, so the behavior is as for recoverable TS queues that are owned by users. If the automatic start follows a normal shutdown, autoinstalled terminals are not restored, and this information is not required.                                                                                                                                  |

## Effect of an unavailable file manager at various stages

Table 33 shows the effect of an unavailable file manager at various stages of CICS processing:

*Table 33. Effect of an unavailable file manager at various stages of CICS processing*

| File manager holding .... is unavailable                                           | A transaction requires the resource prior to syncpoint                                                                                                                                                                                                                                                                                                                                                                                                                                                                | A transaction requires the resource during syncpoint                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| One or more data files.                                                            | Depending on previous processing in the LUW, either a condition code is raised and EIBRCODE is set to the file manager error, or an abend occurs. Message ERZ022012E is logged. CICS continues, but files that are on the file manager become unavailable until it returns.                                                                                                                                                                                                                                           | All two-phase commit activity here is the responsibility of the file manager. The transaction can commit or abort depending on the timing of the unavailability during the two-phase commit. CICS does not mark the file as unavailable or log any messages. CICS continues, but the transaction can possibly block. (Nonrecoverable files are not involved at all in the two-phase commit.) |
| Protected locally queued transactions [START PROTECT NOCHECKs are held on a file]. | This means that a START PROTECT NOCHECK to start a transaction on a remote region is being executed, but the remote region is not available, and the TD entry for the transaction shows that the request is to be locally queued. The semantics of the NOCHECK option mean that the unavailability of the file manager is not returned to the START command, but message ERZ061005E is logged, and the LUW rolls back at the syncpoint. CICS continues, but protected local queueing becomes temporarily unavailable. | All two-phase commit activity is the responsibility of the file manager. The transaction can commit or abort depending on the timing of the unavailability during the two-phase commit. CICS does not log any messages. CICS continues, but the transaction can possibly block.                                                                                                              |
| Nonprotected locally queued transactions [START NOCHECKs are held on a file].      | As for protected locally queued transactions (except that the PROTECT option is absent, and therefore the LUW does not roll back). The START request is ignored.                                                                                                                                                                                                                                                                                                                                                      | Not applicable; this START is not recoverable, and therefore does not participate in two-phase commit.                                                                                                                                                                                                                                                                                       |
| Logically recoverable transient data queues.                                       | Depending on previous processing in the LUW, either a condition code is raised and EIBRCODE is set to the file manager error, or an abend occurs. Message ERZ022012E is logged. CICS continues, but logically recoverable transient data queues become unavailable until the file manager returns.                                                                                                                                                                                                                    | The transaction can commit or abort depending on the timing of the unavailability during the two-phase commit. CICS continues, but the transaction can possibly block.                                                                                                                                                                                                                       |



Table 33. Effect of an unavailable file manager at various stages of CICS processing (continued)

| File manager holding .... is unavailable                                  | A transaction requires the resource prior to syncpoint                                                                                                                                                                                                                                      | A transaction requires the resource during syncpoint                                                                                                                   |
|---------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Physically recoverable transient data queues.                             | As for logically recoverable transient data queues, except message ERZ022011E is logged.                                                                                                                                                                                                    | Not applicable; the file manager processing is nontransactional.                                                                                                       |
| Nonrecoverable transient data queues.                                     | As for logically recoverable transient data queues.                                                                                                                                                                                                                                         | Not applicable; the transient data queue is nonrecoverable and therefore does not participate in two-phase commit.                                                     |
| Recoverable temporary storage queues that are owned by users.             | Depending on previous processing in the LUW, either a condition code is raised and EIBRCODE is set to the file manager error, or an abend occurs. Message ERZ023097E is logged. CICS continues, but recoverable temporary storage queues become unavailable until the file manager returns. | The transaction can commit or abort depending on the timing of the unavailability during the two-phase commit. CICS continues, but the transaction can possibly block. |
| Nonrecoverable temporary storage queues that are owned by users.          | As for recoverable temporary storage queues that are owned by users.                                                                                                                                                                                                                        | Not applicable; the temporary storage queue is nonrecoverable and therefore does not participate in two-phase commit.                                                  |
| Recoverable temporary storage queues that are owned by timer services.    | Depending on previous processing in the LUW, either a condition code is raised and EIBRCODE is set to the file manager error, or an abend occurs. Message ERZ007033E is logged. CICS continues, but START PROTECTs become unavailable until the file manager returns.                       | The transaction can commit or abort depending on the timing of the unavailability during the two-phase commit. CICS continues, but the transaction can possibly block. |
| Nonrecoverable temporary storage queues that are owned by timer services. | As for recoverable temporary storage queues that are owned by timer services.                                                                                                                                                                                                               | Not applicable; the START is nonprotected and therefore does not participate in two-phase commit.                                                                      |
| Recoverable temporary storage queues that are owned by terminal services. | Not applicable; CICS accesses the temporary storage queue only when a terminal signs on and off, at startup, and at shutdown.                                                                                                                                                               | Not applicable.                                                                                                                                                        |

## Effect of other unavailable file managers

Table 34 on page 206 shows the effect of other unavailable file managers at various stages of CICS processing:

Table 34. Effect of unavailability of other file managers

| Unavailable file manager | Effect of absence at startup (cold and automatic)                                                                                                                                                                                                                                                                                       | Effect of absence during normal running                                                                                                                                                                                                                            |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PPC Gateway server.      | CICS starts, but SNA conversations that are using that gateway are not possible. The administrator can subsequently start the gateway, delete its definition from the CICS runtime, then install it again, thereby reconfiguring the gateway. See the <i>TXSeries for Multiplatforms Intercommunication Guide</i> for more information. | Currently running SNA conversations terminate abnormally. You can simply bring the gateway back up again and CICS continues to use it without administrator involvement. See the <i>TXSeries for Multiplatforms Intercommunication Guide</i> for more information. |

The file manager during normal region shutdown must be available if outstanding nonprotected local start requests exist. These requests are held in main storage while the region is running but are transferred to the file manager at shutdown so that they can be retrieved on a subsequent autostart. If the file manager is not available at this time, message ERZ023102E is logged and the region is terminated abnormally.

---

## Chapter 17. Shutting down a CICS system

This chapter describes the CICS shutdown processes and procedures. “The shutdown sequence” describes the sequence in which the file manager and the region are shut down. “CICS shutdown processes” describes the types of shutdowns that can occur with your CICS system, and “Shutting down CICS by using `cicscp`” on page 210 describes the procedures for shutting down the Structured File Server (SFS) server and the region.

---

### The shutdown sequence

The region relies on the underlying components for operation. Therefore, do not shut down those underlying components until the region is shut down. Use the following sequence for shutdown:

1. The region
2. The PPC Gateway server (if one is running)
3. The SFS server

The CICS control program **`cicscp stop all`** command shuts down the components in the correct sequence. See the *TXSeries for Multiplatforms Administration Reference* for more information.

To shut down an SFS server, see “Shutting down an SFS server by using the `cicssfssshut` command” on page 282 or “Shutting down an SFS server by using `cicscp stop sfs_server`” on page 211. For information about shutting down the PPC Gateway server, see the *TXSeries for Multiplatforms Intercommunication Guide*.

---

### CICS shutdown processes

Administrators can specify three types of shutdown:

- A normal shutdown (default shutdown type)
- An immediate shutdown
- A cancel (forced abnormal) shutdown

CICS can also be shut down by a software-initiated process. An uncontrolled shutdown can be caused by a power loss or other system failures.

**Note:** If CICS waits indefinitely during shutdown and no user processes are outstanding, use the **`cicsnotify`** command, as described in the *TXSeries for Multiplatforms Administration Reference*. If this does not help, contact your support organization.

#### Normal shutdown

This section describes the process of normal shutdown, which occurs in *quiesce stages*:

1. The first quiesce stage of termination:

All terminals are active, all CICS facilities are available, and CICS runs tasks and programs as follows:

- Tasks that are already running are allowed to complete.
- Tasks that are to be automatically initiated are allowed to complete, provided that they can start before the second quiesce stage.

- CICS runs any user-written programs that are listed in the first shutdown program list that is held in the Region Definitions (RD).
- CICS allows a new transaction that is requested as a result of terminal input to start only if the transaction has been flagged in the Transaction Definitions (TD) as being allowed to start at shutdown.

**Note:** When you install the region, CICS initially flags the CICS-supplied transactions CEMT (runtime resource management) and CSSF (signoff) as being startable during shutdown.

The first quiesce stage is complete when the last of the programs that are in the first shutdown program list has run and all user tasks are complete.

**Note:** Long-running tasks (such as conversational tasks) must terminate before CICS termination can proceed.

2. The second quiesce stage of termination:

- Terminals are not active.
- CICS does not allow any new transactions to start.
- CICS runs any user-written programs that are listed in the second shutdown program list that is held in the Region Definitions (RD). These programs cannot communicate with terminals or make any request that causes a new task to start.

The second quiesce stage ends when the last of the programs that are listed in the second shutdown program list has finished running.

- CICS writes statistics to the statistics file that are defined in the RD.
- CICS closes all files.
- CICS writes information to the restart file that is to be used for a subsequent automatic start.
- CICS stops running.

If you have set the **SysDump** attribute in the RD to **yes**, CICS generates a system dump. CICS does not always complete the dump process. If **SysDump** is permanently set to **yes**, CICS always produces a dump at shutdown. These dumps are very large, and a dump is not normally required at shutdown.

## Immediate shutdown

When you request an immediate shutdown of CICS, processing differs from a normal shutdown in the following important ways:

- CICS does not wait for user tasks to complete.
- CICS does not run any of the programs that are in the shutdown program lists that are defined in the Region Definitions (RD).
- CICS does not write information to the restart file.

**Note:** CICS does not wait for user tasks to complete, but forcibly terminates running tasks. CICS runs dynamic transaction backout on tasks that are in the middle of a transaction when the tasks are terminated, then shuts down resources.

If you have set the **SysDump** attribute in the RD to **yes**, CICS attempts to generate a system dump. CICS does not always complete the dump process.

For information about how to restore CICS after an abnormal shutdown, see Chapter 20, “Understanding CICS backup and recovery,” on page 239. See also “Shutting down CICS by using cicscp” on page 210 and “Shutting down the region from the command line” on page 281.

## Cancel (forced abnormal) shutdown

As described in “Shutting down CICS by using cicscp” on page 210 and “Shutting down the region from the command line” on page 281, when you request a cancel (forced abnormal) shutdown of CICS, the region is forcibly shut down with no shutdown processing. Use this only if the system stalls during shutdown, or you cannot purge tasks and shut down normally.

If you use a cancel (forced abnormal) shutdown, you must also use the **cicsnotify** command to release any resources that have not been deallocated, and to delete the region lock file.

For information about how to restore CICS after an abnormal shutdown, see Chapter 20, “Understanding CICS backup and recovery,” on page 239.

**Note:** The CICS log files are not cleared unless a region is shut down normally. Repeated abnormal shutdowns can increase the size of the log file.

## Software-initiated shutdown

This type of shutdown occurs when CICS detects that its internal data structures have been overwritten with invalid data, or when some external event, such as sending the SIGKILL signal, causes a crucial CICS process to be destroyed.

The following occurs when software causes CICS termination:

- User tasks are not always completed.
- Shutdown programs listed in the Region Definitions (RD) are not run.
- Information is not written to the restart file.
- A dump can be taken if the **SysDump** attribute in the RD is set to **yes**.
- Open files are not always closed.

For information about how to restore CICS after an abnormal shutdown, see Chapter 20, “Understanding CICS backup and recovery,” on page 239.

## Uncontrolled shutdown

The following conditions can cause an uncontrolled termination of CICS:

- Power loss
- Processor check
- Operating system error

In each case, CICS cannot perform any shutdown processing. In particular, CICS does not write any information to the restart file.

For information about how to restore CICS after an abnormal shutdown, see Chapter 20, “Understanding CICS backup and recovery,” on page 239.

## Statistics produced at shutdown

CICS management modules control how events are managed within CICS. As events occur, CICS produces information that is available to you in the form of system statistics. The statistics show how CICS manages the resources under its control. These resources include files, journals, transactions, programs, and tasks.

CICS produces shutdown statistics at every normal shutdown and immediate shutdown. Shutdown statistics report system activity between the preceding startup of CICS and its shutdown. You can use shutdown statistics for noting trends and trouble spots. You can use the **cicssfnt** program to print this report, but note that **cicssfnt** is a sample program that you modify for your own needs.

For CICS to record shutdown statistics, specify a CICS-defined file or a user-defined file in the Region Definitions (RD). This ensures that CICS has a place to send the recorded data. CICS sends shutdown statistics to this file.

## Printing a software-initiated shutdown dump

Most software-initiated terminations produce information about the termination that can help you handle the problem. The sources of information are as follows:

- **console.nnnnnnn**
- **CSMT.out** file
- **symrecs** file
- Dump file

You can format and print the dump file by using the **cicsdfnt** offline utility program. This program takes dump files and formats them by using the format descriptions that are held in the dump file control block and the arguments that are provided to the program.

If you cannot determine the cause of the software-initiated termination from the dump information, contact your support organization.

---

## Shutting down CICS by using **cicscp**

This section describes how to shut down a CICS system by using **cicscp**. For a description of the full functions of the **cicscp** commands, see the *TXSeries for Multiplatforms Administration Reference*. For alternative procedures, see Appendix A, "Using the AIX System Management Interface Tool (SMIT)," on page 259 or Appendix B, "Using CICS resource management commands for system management," on page 269.

## Shutting down the region by using **cicscp stop region**

The region relies on the underlying components for operation. Therefore, do not shut down the file manager until the region is shut down.

Normally, you do not shut down the file manager because the region can be stopped and started without stopping this component. This is especially important when an SFS server is shared with other regions or non-CICS applications.

**Note:** You can keep track of the region shutdown sequence by opening another window and issuing the command:

```
cicstail -r region_Name
```

This will display the current console and CSMT files for the region, and all lines that are added to these files as the region shuts down.

1. Log in with a user ID that has root privileges and is set up for an administrator, as described in Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.
2. Use this command to shut down the region:

```
cicscp -v stop region regionName
```

where *regionName* is the name of the region.

## Shutting down an SFS server by using `cicscp stop sfs_server`

Use this procedure if you are going to use the `cicscp -v stop sfs_server` command to shut down an SFS server:

**Attention:** Do not shut down the SFS server until all regions that use the SFS server have first been shut down.

1. Log in with a user ID that has root privileges and is set up for an administrator, as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Enter the command:

```
cicscp -v stop sfs_server /./cics/sfs/serverId
```

where *serverId* is the base name for this server.

## Shutting down CICS by using `cicscp stop all`

Use this procedure if you are going to use the `cicscp stop all` command to shut down all CICS objects on the local host.

1. Log in with a user ID that has root privileges and is set up for an administrator, as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Enter the command:

```
cicscp -v stop all
```

---

## Shutting down the region by using CEMT PERFORM SHUTDOWN

You can bring a CICS region down through Run Time Resource Management Transaction (CEMT) by using the CEMT PERFORM SHUTDOWN command on the CICS Terminal. It enables the System Administrator to shutdown the region from the CICS Terminal.

If CEMT PERFORM has been specified, the terminal displays two options:

- SHUTDOWN
- SNAp

The administrator selects SHUTDOWN to Shutdown the CICS region.

The CEMT PERFORM SHUTDOWN command gives two types of shutdown from which an administrator can choose the one that suits current requirements:

- Normal
- Immediate

Normal is the default. When a CICS administrator issues a CEMT PERFORM SHUTDOWN command, CICS tries to shutdown normally. If an immediate shutdown of the region is required, the CICS Administrator must give the CEMT PERFORM SHUTDOWN IMMEDIATE command on the CICS terminal.



After issuing the CEMT PERFORM SHUTDOWN command, the administrator must press the END AID key on the CEMT panel to complete the shutdown process.

---

## Shutting down the region by using EXEC CICS PERFORM SHUTDOWN

EXEC CICS PERFORM SHUTDOWN is a System Programming Interface (SPI) that allows the user-written system programs to shutdown the CICS region. The main purpose of EXEC CICS PERFORM SHUTDOWN SPI is to enable system programmers to write system application programs that can administer the system.

**Note:** Do not use EXEC CICS PERFORM SHUTDOWN SPI inside normal application programs (business applications).

See the *TXSeries for Multiplatforms Application Programming Reference* for information about the options and syntax of the EXEC CICS PERFORM SHUTDOWN SPI.

---

## Chapter 18. Monitoring your CICS system

CICS provides statistics, monitoring, and tracing tools that you can use to get information about the running of a CICS system. You can use the offline utility programs **cicssfnt**, **cicsmfnt**, and **cicstfnt** to format the information that these tools provide. You can then analyze the formatted data to provide information about any resource contention or other problems that can affect the performance of your CICS system.

**Note:** The sample program **cicsmfnt** formats only a subset of the monitoring data. The sample program **cicssfnt** provides a basic listing of the statistics.

You can tailor these tools and offline programs to provide information at specific event points or at specific time intervals in the running of the CICS system. You can also tailor them to specify the types of data about which CICS is to provide the information.

“Collecting statistics” describes the types of statistics that you can collect and the methods that you can use to collect them.

“Setting up Journal Definitions (JD)” on page 217 describes how to enable the use of journals with your CICS system.

“Monitoring performance” on page 219 talks about the methods that you can use to monitor your CICS system.

“Tracing CICS execution performance” on page 221 describes how to use auxiliary tracing.

---

### Collecting statistics

This section describes how to use the CICS statistics facility to collect statistics from a CICS system, and how to format a report that is based on those statistics.

#### Information collected for CICS statistics

Use the CICS statistics facility to collect information about the management of resources such as files, queues, journals, and transactions. For example, collect information about the number of times a file is read, the number of unauthorized access attempts to a specific journal, or the total number of transactions that are started at a particular time.

#### Categories of CICS statistical information

CICS statistical information can be collected for the following categories:

- **DUMP - Dump Statistics:** Records the number of dumps that are written and the number of write errors that occurred while dumps were taken.
- **FILE - File Statistics:** Records input and output activity of application requests against the files.
- **INTERSYSTEM - Intersystem Communication statistics:** Records intersystem communication statistics.
- **JOURNAL - Journal statistics:** Provides statistical information for every journal that is defined to a region.

- **PROGRAM - Program statistics:** Provides statistical information for all programs that are defined in the Program Definitions (PD).
- **RUNTIMEDB - Runtime Database statistics:** Records statistics about the use of the runtime database and the collision characteristics of the supporting hash table.
- **STORAGE - Storage statistics:** Provides information about the region pool, task-shared pool, and task-private pool.
- **TASK - Task statistics:** Provides information that relates to tasks.
- **TDQUEUE - Transient Data statistics:** Provides statistical information about each transient data queue that is defined in the Transient Data Definitions (TDD). It also accumulates statistics for all TD requests since region startup.
- **TERMINAL - Terminal statistics:** Provides information about terminals that are defined in Terminal Definitions (WD). This information is important; it shows how users are loading the system.
- **TRANSACTION - Transaction statistics:** Provides information about all transactions that are defined in the Transaction Definitions (TD).
- **TSQUEUE - Temporary Storage statistics:** Provides information about temporary storage queues that are defined in the Temporary Storage Definitions (TSD).
- **UNITOFWORK - Logical Unit of Work (LUW) statistics:** Provides accumulated statistics that relate to LUWs.

## Types of statistics

For each category of information, four types of information can be collected; their descriptions follow. For information about how to generate each type of statistics, see “How to collect statistics and format a report.”

- *Interval statistics* indicate system activity for a specified time interval. At the end of each interval, CICS writes the collected statistics to disk and resets the statistics values to zero. Interval statistics are useful for overall analysis of specific time periods or peak activity periods. By specifying short intervals within a specified period, you can look at a particular activity in more detail and expose potential problems that are hidden by larger amounts of statistical data.
- *End-of-day statistics* report system activity between the end of the previous interval and the following:
  - The end-of-day expiry time
  - When CICS quiesces (normal shutdown)
  - When CICS terminates (immediate shutdown)

You can use end-of-day statistics for noting trends and trouble spots. Because this report is generated after every normal CICS termination, examine it regularly.

- *Requested statistics* record the state of a resource immediately, instead of waiting for the current interval to expire. Requested statistics are useful for analyzing temporary or short-term problems.
- *Unsolicited statistics* are gathered by CICS automatically for dynamically allocated and deallocated resources. These are statistics that are written about resources that are about to be deleted and whose statistics will otherwise be lost.

## How to collect statistics and format a report

The following Region Definitions (RD) attributes and CICS API commands are used to record statistics:

- The RD **StatFile** attribute specifies the pathname and file to which statistical data is sent. All statistics are written to this file except those from the EXEC CICS COLLECT STATISTICS command, which provides statistics information for application programs.
- The RD **StatsRecord** attribute must be set to **yes** for interval or unsolicited statistics to be created. CICS ignores this attribute when it produces end-of-day and requested statistics.
- The EXEC CICS SET STATISTICS command is used within an application program. It sets the time at which CICS collects the end-of-day and interval statistics. This is not applicable for requested or unsolicited statistics.
- The EXEC CICS PERFORM STATISTICS RECORD command generates detailed requested statistics and resets the statistics values to zero. This command sends output to the file that is specified by the RD **StatFile** attribute. This is not applicable for end-of-day, interval, or unsolicited statistics.
- The EXEC CICS COLLECT STATISTICS command gives an application program access to the statistics by returning to the calling application the current statistics for a single resource. After CICS writes the data to the file that is specified by the RD **StatFile** attribute, you cannot access the statistics with the EXEC CICS COLLECT STATISTICS command.

You can issue the API commands by using the CECI transaction or from within an application. See the *TXSeries for Multiplatforms Application Programming Reference* for more information on INQUIRE and SET commands.

The statistical output is unformatted. You can format and print this output by using the supplied offline utility **cicssfnt**, or you can write your own program to report and analyze the statistics.

### How to generate end-of-day statistics

1. Ensure that the **StatFile** attribute in the RD is set.  
These statistics are produced regardless of the setting of the **StatsRecord** attribute in the RD.
2. Use the EXEC CICS SET STATISTICS command within an application program to set the time at which CICS collects the end-of-day statistics.

### How to generate requested statistics

To generate requested statistics:

1. Ensure that the **StatFile** attribute in the RD is set.
2. Use the EXEC CICS PERFORM STATISTICS RECORD command to request all or specific resource statistics that are currently held. For example, to request only terminal statistics, use the EXEC CICS PERFORM STATISTICS RECORD TERMINAL command to send output to the file that is specified by the **StatFile** attribute in the RD.

You can also use the EXEC CICS PERFORM STATISTICS RECORD ALL RESET NOW command to reset the statistics values to zero.

The EXEC CICS COLLECT STATISTICS command returns current statistics for a single named resource or the global summary statistics for a named resource type to the invoking application. For example, you can get the statistics for all programs in your CICS system, or for only a single program.

## How to generate interval statistics

Use the interval statistics facility to obtain statistics automatically for time intervals that you specify. This facility accumulates information for the time interval, and CICS clears this information at the end of each interval. You can specify an interval that ranges from one minute to 24 hours.

CICS starts recording interval statistics automatically at startup, with a default interval of three hours, and an end-of-day time of 00:00.

To generate interval statistics:

1. Ensure that the **StatFile** attribute in the RD is set. This attribute specifies the file to which CICS writes statistical output.
2. Ensure that the **StatsRecord** attribute in the RD is set to **yes**.
3. Use the EXEC CICS SET STATISTICS command to control when and what is recorded. For example, if CICS fails, it attempts to write end-of-day statistics for the run, but it can fail to do so. If CICS cannot write end-of-day statistics, you can use interval statistics to request all statistics periodically, for example, every hour or half hour, and use this information to analyze the reasons for the failure.

Alternatively, you can request the complete statistics at intervals by running an EXEC CICS PERFORM STATISTICS RECORD command manually at intervals during the day. Normal shutdown puts a final set of statistics on the file; in all other cases, you get the last available set.

## How to generate unsolicited statistics

Do the following to generate unsolicited statistics:

1. Ensure that the **StatFile** attribute in the RD is set.
2. Ensure that the **StatsRecord** attribute in the RD is set to **yes**.

CICS sends the generated statistical data to the specified pathname. The statistical output is unformatted.

## Inhibiting the collection of interval and unsolicited statistics

You can inhibit the production of interval and unsolicited statistics by setting the **StatsRecord** attribute in the RD to **no** or by setting the RECORDING parameter in the SET STATISTICS command to OFF. The end-of-day and requested statistics are still generated, regardless of the setting of this attribute.

CICS continues to record data until you cancel the operation by using the EXEC CICS SET STATISTICS command.

## Generating statistics reports

You can use **cicssfnt** to generate two different types of statistics report. One is an interval report, which is detailed, and the other is a summary report. See the *TXSeries for Multiplatforms Administration Reference* for more information.

Both reports use all the statistics collections that are contained in the file. Therefore, depending on when you generate reports and when you last cleared the file, you can produce reports that cover an hour, a week, or any other period of time that you want. It is recommended that you limit the summary period to a year or less, because summary data can become very large.

- The *Interval report* contains the detailed statistics for each individual resource and the summary statistics for each resource category. For example, it contains information about CSMT in the Transient Data statistics, information about CECI in the Transaction statistics, and information about resource categories such as

Transient Data and Intersystem Communication. The interval report displays all the statistics collections that are contained in the statistics file. These statistics collections are the interval, end-of-day, requested, and unsolicited collections.

- The *Summary report* contains only the summary statistics for each report category, for example Transient Data and Intersystem Communication, taken from all the collections that are contained in the statistics file. These statistics collections are the interval, end-of-day, requested, and unsolicited collections. The summary report facility also produces reports that average the resource category summary statistics and give the totals, peaks, and averages. These reports are generated only for end-of-day and interval collections.

### Headings produced by `cicssfnt`

The following headings are produced by `cicssfnt`:

- **REQUESTED\_RESET Statistics Collection:** Statistics that are recorded for all resource types, as requested with the following command:  
`EXEC CICS PERFORM STATS REC ALL RESETNOW`  
When `RESETNOW` is used, statistics counters are reset for all resources.
- **REQUESTED Statistics Collection:** Statistics that are recorded for all resource types, as requested with the following command:  
`EXEC CICS PERFORM STATS REC ALL`
- **INTERVAL Statistics Collection:** Statistics that are taken when the interval time has expired. This section contains detailed and summary statistics.
- **ENDOFDAY Statistics Collection:** Statistics that are taken when the end-of-day time has been reached. This section contains summary statistics.
- **ENDOFREGION Statistics Collection:** Statistics that are taken when region is shut down. This section contains summary statistics.
- **UNSOLICITED Statistics Collection:** Statistics that are taken for dynamically allocated and deallocated resources. This section contains summary statistics.

---

## Setting up Journal Definitions (JD)

A journal is a set of special-purpose sequential files. Journals can contain any and all data that the application program needs to enable subsequent reconstruction of events or data changes. For example, a journal can act as an audit trail, a change-file of database updates and additions, or a record of transactions that are passing through the system (often called a *log*). Each journal can be written from any task.

The various Journal Definitions (JD) and their attributes are described in the *TXSeries for Multiplatforms Administration Reference*. Of particular interest are the attributes **DiskA** and **DiskB**. These attributes specify the name of the file (in *regionName/data*) that holds the journal. Both files must be writable, although only one file is used at a time. The file that is in use is set to **current** (see the **DiskAStatus** and **DiskBStatus** attributes). You can request to switch between **DiskA** and the alternative file in **DiskB** by using the `EXEC CICS SET JOURNAL` or `CEMT SET JOURNALNUM` commands.

CICS does not empty user journals, so you must clear them when they are no longer required. If CICS is offline, delete the two operating system files that are defined as **DiskA** and **DiskB**.

## Configuring the Journal Definitions (JD)

The following command examples show how to add, change, delete, or view the Journal Definitions (JD) that are in the permanent and runtime databases. Full descriptions of Journal Definitions (JD) and their attributes, including default values, are given in the *TXSeries for Multiplatforms Administration Reference*.

- To add a new JD:

This example shows how to add a JD entry known to CICS as **01** and uses the following files for the journal log:

- /var/cics\_regions/regionA/data/J1PRIME
- /var/cics\_regions/regionA/data/J1ALT

The defaults are assumed for all other JD attributes.

```
cicsadd -c jd -r regionA -P 01 DiskA=J1PRIME DiskB=J1ALT
```

Because the default setting for **DiskAStatus** is **current**, and the default setting for **DiskBStatus** is **ready**, /var/cics\_regions/regionA/data/J1PRIME is used for the journal log until the settings are changed.

If you want to add the definition to the runtime and permanent databases simultaneously, ensure that the region is running and use the **-B** (both permanent and runtime) option, as follows:

```
cicsadd -c jd -r regionA -B 01 DiskA=J1PRIME DiskB=J1ALT
```

- To change an existing JD:

This example shows how to change a JD entry so that it is not installed into the runtime the next time the region is cold started.

```
cicsupdate -c jd -r regionA -P 01 ActivateOnStartup=no GroupName=''
```

To install the definition into the runtime, see “Installing a resource definition from the permanent database” on page 118.

**Note:** If the **GroupName** attribute contains a group name that is included in the region’s **Groups** list, the JD entry is installed regardless of the **ActivateOnStartup** value.

- To delete a JD entry:

This example shows how to delete a JD entry.

```
cicsdelete -c jd -r regionA -P 01
```

To delete the entry from the runtime, see “Deleting a resource definition only from the runtime database” on page 118.

- To view a JD entry:

This example shows how to view a JD entry:

```
cicsget -c jd -r regionA 01
```

**On the AIX platform.** See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about how to use SMIT to define resources to the region on the AIX platform.

### Working with JD in the runtime database

To work with the Journal Definitions (JD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change Journal Definitions (JD) while the region is running. Any changes that you make by using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions on the next restart. CEMT lets you:

- View the runtime Journal Definitions (JD)



- Open or close runtime journals
- Switch the disk file or a runtime journal
- View the status of the runtime journal disk files
- Empty a journal

For more information, see the *TXSeries for Multiplatforms Administration Reference*. To write CICS application programs to manage the status or operation of journals in the runtime, see the *TXSeries for Multiplatforms Application Programming Reference*.

---

## Monitoring performance

You can collect two types of information:

- Standard CICS information
- Information from CICS application programs

### Standard CICS information

The information that is gathered by CICS is mainly performance related, but information that relates to exception conditions is also available.

Transaction-level data records (transaction performance) are written to the transient data queue that is specified by the **TDQ** attribute in the Monitoring Definitions (MD). These records contain information that relates to the use of resources by individual tasks. These are recorded for each task when the task terminates.

You can enable system performance-class monitoring by invoking the CEMT SET MONITOR ON transaction or by setting the **MonitorStatus** attribute in the MD to **yes**. See “Identifying exceptions to when resource definitions can be changed” on page 123 in Chapter 8, “Working with CICS resource definitions,” on page 115 for information about using CICS resource definitions online (RDO) commands.

**On the AIX platform.** See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about using SMIT.

### Information from CICS application programs

In addition to collecting standard CICS information, the CICS monitoring facility allows application programs to pass to the monitoring facility data that is in user fields. This data can be used for accounting and transaction-level performance evaluation. A variable number of user fields (counts, time periods, and character strings) are permitted in transaction-level performance records.

If you want to gather information in addition to that provided by the CICS monitoring facility, you can select points in an application program for special trace commands that collect the monitoring data. These trace commands are called *event-monitoring points*. You can start and stop user-defined clocks and update user counts at these points.

To support the collection of user-monitoring information, you must provide a user-defined application program to perform the necessary actions at each event-monitoring point. This application program obtains the required information and returns the user-monitoring information so that at the end of the transaction, CICS can append it to the performance-monitoring data written to the transient data queue.

CICS provides additional user-defined counter and clock fields that enable you to count the number of times a particular event occurs within an application or

within the total CICS system. They also enable you to time the interval between two events within an application or within the CICS system.

See the *TXSeries for Multiplatforms Application Programming Reference* for more information.

## What you must define in the Monitoring Definitions (MD)

The following attributes are defined in the MD:

- The **MonitorStatus** attribute specifies whether monitoring is active or not.
- The **TDQ** attribute specifies the transient data queue that CICS uses to collect the required monitoring data. If **TDQ** contains a null queue name, then monitoring occurs, but nothing is written.
- The **UseMonitorModule** attribute specifies the user-defined program that collects monitoring data when the application programmer specifies user-defined event-monitoring points.
- The **Include** and **Exclude** attributes specify which classes of monitoring data are collected and which are not. You can specify the value of the **Include** and **Exclude** attributes as **ALL** to include or exclude all attributes for monitoring.

**Note:** No maximum is defined for the number of fields that you can keep for use with event-monitoring points, so you can keep an unlimited amount of data. However, if the overall size of the user (event-monitoring point) area and the performance area is greater than the maximum record size of the transient data queue, monitoring for that task is disabled. The size of the performance area depends on which monitoring fields are included or excluded.

For a list of monitoring fields and their associated groups, see the *TXSeries for Multiplatforms Administration Reference*.

## Setting up Monitoring Definitions (MD)

If an application program uses EXEC CICS ENTER statements to write trace entries, Monitoring Definitions (MD) are required. The *TXSeries for Multiplatforms Administration Reference* describes the MD attributes.

To work with the Monitoring Definitions (MD) in the runtime database, use the Runtime Resource Management (CEMT) transaction. This transaction allows you to examine and change Monitoring Definitions (MD) while the region is running. Any changes that you make by using CEMT apply only to the current session of CICS; they are replaced with the permanent resource definitions on the next restart. Use CEMT to do the following:

- Find out whether monitoring is switched on in the runtime
- Switch the monitoring facility on or off in the runtime

For more information, see the *TXSeries for Multiplatforms Administration Reference*.

## How to control CICS monitoring

To use monitoring:

- Ensure that monitoring is defined for the system and that the region is cold started after any changes are made to the resource definitions.
- Ensure that monitoring is turned on as follows:
  - At system initialization. To specify whether monitoring is on or off when CICS is initialized, use the **cicsupdate** command to set the **MonitorStatus**

attribute in the permanent MD to **yes** or **no**. See “Identifying exceptions to when resource definitions can be changed” on page 123 for information about using CICS resource definitions online (RDO) commands.

**On the AIX platform.** See Appendix A, “Using the AIX System Management Interface Tool (SMIT),” on page 259 for information about using SMIT.

- During runtime. Use the CEMT SET MONITOR transaction to turn the CICS monitoring facility on or off at any time after CICS is initialized.

To specify the monitoring data that is to be reported, use CICS resource management commands to change the **Include** and **Exclude** attributes in the MD. These changes take effect the next time the region is cold started.

## Processing of CICS monitoring-facility output

CICS provides an offline utility called the Monitoring Data Formatter (**cicsmfmt**) in the form of an application program. This utility reads, formats, and prints the monitoring transient data queue. It is intended as a sample program that you can use as a skeleton if you need to write your own program to analyze the data set.

Alternatively, you can write your own application program to process output from the CICS monitoring facility. See the *TXSeries for Multiplatforms Administration Reference* for more information.

---

## Tracing CICS execution performance

External trace is the trace facility that is provided by the operating system. It exists on the AIX, HP-UX, and Solaris platforms. This trace facility is called the Event Performance Trace Facility (EPTF). It can collect trace from all operating system products at once. It provides nanosecond time stamps so that you can evaluate which products or processes are taking the processor time. For a CICS region to write to the EPTF, you must set the **ExternalTrace** attribute in the Region Definitions (RD). For more information on how to use the EPTF, see the information that is provided for your operating system.

Trace information can be recorded in the main memory buffer, but in most cases the main memory buffer is relatively small and entries can be overwritten. To get a trace of CICS activity in which CICS does not overwrite trace entries, you can use a further method of recording known as *auxiliary trace*. CICS records the trace entries additionally in a sequential file. The format of trace entries in the auxiliary trace file is identical to that for main memory buffer.

You can use one or two auxiliary trace files. If two files are defined and the first file becomes full, you can switch to the second file while CICS is running.

Because of the volume of output it produces, activate auxiliary tracing only for limited periods of time, for example, when you need to repeat a problem condition because the main memory buffer tracing does not provide enough information.

You can activate main memory buffer and auxiliary tracing independently and in either sequence. However, CICS does not record auxiliary trace entries unless the main memory buffer tracing is active at the same time.

CICS automatically terminates auxiliary trace output when it reaches the end of the file allocation. You can direct the output to either of two files, so that output

can continue to one file while CICS prints the other file. However, you have to switch between the files manually, and CICS can lose some traces during this process.

When you have completed auxiliary tracing and after CICS has closed the trace file, you can format and print the information with the help of the trace formatter (**cicstfmt**) that provides functions to read, format, and print the trace data into standard output. Invoke **cicstfmt** directly from the operating system.

## General comments on using auxiliary tracing

CICS auxiliary tracing is a powerful aid for investigating CICS performance problems. However, activating the tracing can involve appreciable extra overhead, principally because it involves extra input and output operations compared with using the CICS main memory buffer (which auxiliary tracing requires as a prerequisite). As a result, activating auxiliary tracing can worsen problems with performance at peak load. However, the amount of data that you obtain from an auxiliary trace can be helpful in analyzing the problem, so the temporary degradation of the system can still be justified.

Before you plan to get auxiliary trace data, however, examine all other sources of information that are available: CICS data from shutdown statistics, the CICS monitoring facility, and so on.

The *TXSeries for Multiplatforms Problem Determination Guide* provides information about all aspects of the CICS trace facility, including manipulating the two tracing types and the offline utility **cicstfmt**.

---

## Chapter 19. Improving the performance of your CICS system

Achieving an acceptable level of performance is often a difficult task because the user's need to experience a quick response conflicts with these other business needs:

- Validating access to CICS regions and resources
- Achieving a large throughput of CICS tasks
- Giving priority to the most important tasks, terminals, or users
- Backing up data that is affected by CICS tasks
- Staying within the enterprise's hardware budget

"Tuning a CICS system" details the recommended process for making tuning changes to your system.

This chapter also provides information about the tools that you can use to measure your system's performance. See "Measuring performance" on page 224.

"Symptoms and solutions for CICS performance constraints" on page 225 looks at storage and memory warnings and poor response time as symptoms that indicate major system constraints.

"Performance considerations for system administrators" on page 230 looks at the types of performance problems that generally require the attention of system administrators.

"Performance considerations shared by administrators and developers" on page 236 looks at the types of performance problems that require consideration from both the system administrator and the developer.

"Possible effects of changing the system clock" on page 238 discusses the impact on the system of changing the system clock.

---

### Tuning a CICS system

To tune a system, you must:

1. Determine acceptable tuning trade-offs
2. Make changes to the system
3. Review the results of the tuning

#### Determining acceptable tuning trade-offs

A system always has constraints. Tuning often involves decreasing the load on one resource only to increase the load elsewhere, so removing one constraint can create another. Identify which resources can accept an additional load in the system without becoming constraints themselves. Tuning is the art of balancing these trade-offs.

#### Making changes to the system

The next step in the tuning process is to make the system modifications that are intended to improve performance. Consider several points when adjusting the system:

- Do not tune just for the sake of tuning. Tune to relieve identified constraints. Tuning resources that are not the primary cause of performance problems has little or no effect on response time. Also, a highly tuned system can deliver the best performance, but such a system is more prone to *peaks* and probably requires more maintenance than a mildly tuned or standard system.
- In general, tuning needs to address major constraints first, particularly those that have a significant effect on response time. Prioritize tuning actions so that those having the greatest effect are done first. It is possible that a single tuning change can completely solve your performance problem.
- Make one tuning change at a time, and measure its effect. If you make multiple changes simultaneously, it can be difficult to tell which one has the significant effect.
- Continue to monitor constraints during the tuning process. Constraints vary over time, so tuning must be done on a new constraint if the old one is no longer the restricting influence on performance.
- Back up your configuration files before starting the tuning process so that you can easily back out any changes if they cause a problem.

## Reviewing the results of the tuning

Measure performance after each adjustment that you make, and review the results:

- If performance has improved to the point that it meets objectives, no more tuning is required.
- If performance is better but not yet acceptable, investigate further to determine your next action and to check whether the resource you tuned is still a constraint.

If it is not, identify other constraints and tune them accordingly.

---

## Measuring performance

The operating system and CICS provide several tools or facilities. You can use a combination of these to measure your system's performance.

### Performance measurement tools provided by the operating system

- **vmstat**: Reports statistics about virtual memory, disks, traps, and CPU activity. It is used to discover system load activity.
- **iostat**: Reports CPU and input/output statistics for tty devices, disks, and CD-ROMS.
- **monitor** (AIX platform): Starts and stops profiling. See your operating system documentation for more information.
- **trace** (AIX platform): Monitors selected system events such as entry and exit to selected subroutines, kernel routines, and interrupt handles. Events are shown in time sequence and in the context of other events. Can be used to examine who, when, how, and why an event happened. See your operating system documentation for more information.
- **sar**: Records and reports a broad variety of system statistics.
- **netstat**: Records and reports network activity by displaying the contents of various network-related data structures.

See the operating system documentation for more information.

## Performance measurement tools provided by CICS

- **Monitoring:** Collects performance-related information at several predetermined points within CICS. You can also embed monitoring points within a user-defined application program. For instance, you can use the monitoring facility to determine the following information for a particular task:
  - The number of syncpoint requests that are made
  - The total time that is spent waiting for terminal input
  - The elapsed time that is spent waiting for a TCP/IP link

You can use the **cicsmfmt** command to format and print performance records. See “Monitoring performance” on page 219 for more information.

- **Statistics:** Shows how CICS manages the resources under its control. The resources are divided into several categories (such as File, LUW, Terminal), and the statistics are divided into several types (such as Interval, End-of-Day, Requested).

With the statistics facility you can answer questions such as: How often was a particular file updated by file control? How many transactions ran in a region during a particular time period? How many Distributed Programming Link (DPL) function shipping requests were received from remote systems?

You can use the **cicssfnt** command to format an interval report and a summary report. See “How to collect statistics and format a report” on page 214 for more information.

- **Trace:** Shows the flow of control through CICS or an application program. CICS offers access to three types of trace: main memory, auxiliary, and external (the operating system).

You can use the **cicstfmt** command to selectively format and print the contents of the auxiliary trace file. See “Tracing CICS execution performance” on page 221 for more information.

### Viewing CICS statistics with the CSTD transaction

To obtain a snapshot of the current performance of a CICS region, use the CICS-supplied transaction CSTD (described in the *TXSeries for Multiplatforms Administration Reference*). The CSTD transaction gives you access to the more important counters relevant to performance, such as transaction, task, terminal, storage, program, and file statistics.

## Performance measurement tools provided by relational databases

Refer to the DB2 documentation for information about performance-measurement tools. If you use DB2 for queue and file management, read the information that is given in the *DB2: Administration Guide* about tuning and configuring the system.

---

## Symptoms and solutions for CICS performance constraints

Warnings of storage stress conditions, messages indicating memory shortages, and poor response time are common symptoms of major constraints on a CICS system. This section describes these symptoms in detail so that you can recognize them when your system has a performance problem and know how CICS itself attempts to resolve various conditions.



## Symptoms and solutions for storage stress conditions and memory shortages

Storage stress conditions, storage violations, adequate physical memory, and swap space are all issues that affect the performance of your CICS system. This section explores the symptoms of problems that are caused by shortages in storage, memory, and swap space, and it provides suggestions for solving these problems.

### Identifying storage stress conditions

*Storage stress* is the term that CICS uses to indicate that a shortage of free memory exists or is anticipated in the CICS storage pools:

- Region storage pool
- Task-shared storage pools
- Task-private storage pools

CICS indicates a stress condition in a pool by displaying warning messages on the **console.nnnnnnn** file. CICS raises the storage stress condition when the amount of storage that is allocated from a pool has exceeded the pool's stress threshold (for the region and task-shared pools), or when no storage remains (for any pool).

You can configure the storage threshold values (defined as a percentage of the storage pool) by using various attributes in the Region Definitions (RD). A reasonable default for the threshold values is 90 percent of the pool size.

Consider the stress messages an early warning; CICS can then begin to show poor performance characteristics. For example:

- Transactions can abnormally terminate because not enough free storage is available in the region pool.
- Transactions can become suspended while waiting for storage in the task-shared pool to satisfy map or data table requests.
- Transactions can have the NOSTG condition raised from GETMAIN because no storage is available in a task-private pool.
- If not enough paging space is available, transactions can fail to start.

You can use CICS statistics and monitoring facilities to analyze storage usage in different storage pools. From such data, you can detect patterns of use and take action.

The main reasons for storage stress conditions in a CICS region follow:

- The region pool is undersized for the number of resources that are defined and active in the region.

CICS uses the region pool for control blocks. No relationship exists between the number of resources that are active in the region and the amount of region storage that is allocated for control blocks.

- The task-shared pool is undersized for the number of maps and data tables that are to be used in the region.

CICS shares loaded maps and tables across application servers, but a large number of different maps and tables can stress an undersized task-shared pool. CICS uses a hashing mechanism to locate task-shared storage areas and loaded maps and tables. You can configure the size of the hash tables by using the Region Definitions (RD). The two hash tables are **TaskSHNumBuckets** and **LoadDataNumBuckets**. **LoadDataNumBuckets** contains hash values for anything that can be actually loaded; for example, programs and tables. In addition to programs and tables, **TaskSHNumBuckets** contains terminal

input-output areas (TIOA), COMMAREAs, and other storage areas that tasks use. Subsequently, **LoadDataNumBuckets** is a subset of **TaskSHNumBuckets**. For this reason, it is best to keep these two values the same. Then you can use the storage statistics for tuning both of these. See also the descriptions of the **LoadDataNumBuckets** and **TaskSHNumBuckets** attributes in the *TXSeries for Multiplatforms Administration Reference*.

Table sizes that are greater than 512 can be appropriate if the region uses very many maps and tables.

- Application code that consists of many LINK or XCTL programs use large amounts of GETMAIN storage that is never released. Investigate such applications and consider recoding them to use FREEMAIN when the storage is no longer required. Consider redesigning such applications and breaking them into even smaller units.
- The operating system can show symptoms of storage stress when it handles many processes (CICS application servers or other programs).

For more information about tuning various storage conditions, see “Looking at physical memory to improve performance” on page 228 and “Looking at swap space to improve performance” on page 229.

## Identifying storage violations

CICS detects storage violations by using *signature* strings. CICS places a signature string in storage immediately following the storage that is allocated to a calling program. CICS checks the signature string when it releases the associated storage area. The CICS self-consistency checking transaction checks all signature strings in the region and task-shared storage pools.

CICS takes different actions when storage violations occur, depending on which storage pool is affected:

- Task-private pool: CICS checks for storage violations when a program explicitly calls FREEMAIN and during storage reclamation at the end of a task.  
If CICS detects a storage violation, it logs a message to CSMT or **console.nnnnnnn** indicating the address where the storage violation occurred, and recovers the storage areas. When CICS detects a storage violation in FREEMAIN, it raises an A47C abnormal termination code on the task. CICS initiates a transaction dump if the **TransDump** flag is set in the Transaction Definitions (TD) entry for the transaction.
- Task-shared pool: CICS uses its self-consistency transaction to check for storage violations in this pool. If CICS detects a storage violation, CICS logs a message to **console.nnnnnnn**, restores the signature string, and recovers the storage areas. Running transactions and the region continue processing.
- Region pool: CICS uses its self-consistency transaction to check for storage violations in this pool. If CICS detects a storage violation, CICS logs a message to **console.nnnnnnn** and abnormally terminates the region. CICS initiates a system dump if the **SysDump** flag is set in the Region Definitions (RD).

You can use the following to help you to identify the problem areas:

- Messages in **CSMT.out**
- Messages in **console.nnnnnnn**
- Storage statistics
- Transaction dumps
- System dumps

For more on storage violations, see the *TXSeries for Multiplatforms Problem Determination Guide*.

## Looking at physical memory to improve performance

If a system that is running CICS and an SFS server has plenty of memory, the only time that a physical disk is used by a transaction is during syncpoint. During syncpoint, information is forced to disk so that the transaction can be redone if the system fails. All other data, such as the transaction programs themselves, the records in the files, and working storage, is read into physical memory the first time it is required and remains there.

If you have enough physical memory, it is unlikely that disk-operation time will constrain your system. However, practical systems do not normally have this much physical memory, and files under management are usually much larger than the available physical memory. *Swapping* is used to move files temporarily in and out of main memory. When the amount of available physical memory is further reduced, the system goes into *thrashing*. When thrashing occurs, almost all the operating system processes in the system cannot make progress because they are waiting for page faults to be resolved. After a page fault is resolved, the process that is waiting for it makes a small amount of progress but very soon runs into another page fault and is delayed again.

When a system is thrashing, the short-term solution is to reduce the number of application servers. With a reduced load, the system can perform its tasks faster. However the throughput of transactions suffers.

The solution to this problem over the long term involves installing more real storage or redesigning the transaction programs to use less storage. You can also increase the speed or number of disks that are used for paging.

You can tell when your system is paging and when it is thrashing by observing the terminal response time, which is usually slow during paging. If you increase the number of application servers and the system throughput decreases, the system is thrashing.

**Tips on using memory-tuning aids:** You can use some SFS resource attributes and, on the AIX platform, an operating system command, to tune memory storage or the use of memory storage:

- SFS buffers.

The SFS server manages an amount of storage directly. When SFS storage is full, it passes data page-by-page to the operating system. If the SFS buffering is too small, it spends all its time fetching and passing data to the operating system. If the SFS buffering is too large, the operating system performs excessive paging underneath SFS. You can adjust the amount of SFS storage by changing the **BufferPoolSize** attribute in the Structured File Server Definitions (SSD).

- SFS processing threads.

The **OpThreadPoolSize** attribute in the Structured File Server Definitions (SSD) sets the maximum number of file-control requests to an SFS server (on behalf of all application servers in all regions) that can be in progress at any time. Excess requests are queued in the interface between CICS and the SFS server and are dealt with as previous requests complete. A practical initial figure is 10; consider increasing this if the system is idle because it is waiting unnecessarily for SFS threads, or consider reducing this if the system is thrashing.

- **schedtune**.

The operating system process dispatcher has a feature that detects the onset of thrashing and can take action to stop the whole system from slowing down until it stops. The action it takes is to suspend, usually for several tens of seconds, those operating system processes that take a lot of page faults. Suspending a CICS or SFS process can cause a transaction rollback or a transaction program abend, and it causes several tens of seconds of delay to the terminal user. If a user's **cicslterm** process is chosen, response to keystrokes is lost during the suspension, and a transaction program can abend because an EXEC CICS SEND or EXEC CICS CONVERSE call fails.

**On the AIX platform.** You can use the operating system **schedtune** command to override the sensitivity of this detection and the severity of the action taken, but users will still see unacceptably long and variable response times if the system is thrashing, whether or not this facility is tripped. Normally, it is better to take the steps to prevent thrashing rather than try to use **schedtune** to vary the effects of thrashing.

### Looking at swap space to improve performance

It is often recommended that swap space be three times as large as memory. You can define more swap space than this; the only disadvantage is that the disk space so allocated cannot be used for SFS or operating system files. It is more efficient to distribute swap space over multiple disks if large amounts of it are required.

It can be appropriate to define a considerably smaller amount of virtual storage than this for a system that is running only CICS. If large amounts of virtual storage are in use, users' response times will probably suffer.

## Symptoms and solutions for poor response time

Table 35 details the scale that identifies the severity level of response time problems. It explores the symptoms, causes, and solutions that surround each severity level. The scale reports the severity level in decreasing sequence of severity with **1** as the most severe level and **4** as the least severe level.

Table 35. CICS response time checklist

| Level | Symptom                                          | Major causes                                                                                                                                                                                                                                                         | Overall solution                                                                                                                                                                                                                                                                                                                                                                         |
|-------|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | Poor response at all loads for all transactions. | <ul style="list-style-type: none"> <li>High level of paging.</li> <li>Very high usage of major resources.</li> </ul>                                                                                                                                                 | <ul style="list-style-type: none"> <li>Allocate more physical memory. Reevaluate system-resource requirements and redesign system.</li> <li>Check for application errors and resource contention.</li> </ul>                                                                                                                                                                             |
| 2     | Poor response at medium and high loads.          | <ul style="list-style-type: none"> <li>High level of paging.</li> <li>High processor usage.</li> <li>High DB or file usage.</li> <li>High communications network usage.</li> <li>TP or I/O access-method constraint.</li> <li>CICS limit values exceeded.</li> </ul> | <ul style="list-style-type: none"> <li>Allocate more physical memory.</li> <li>Reduce path length or increase processor power.</li> <li>Reorganize files, reduce data transfer, or increase capacity.</li> <li>Reduce data transfer or increase capacity.</li> <li>Increase buffer availability. Change operands, provide more resources, or check for errors in application.</li> </ul> |

Table 35. CICS response time checklist (continued)

| Level | Symptom                                       | Major causes                                                                                                                                                                                                                                                                                                                 | Overall solution                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3     | Poor response for specific transactions only. | <ul style="list-style-type: none"> <li>Identify common characteristics.</li> <li>Lines or terminal usage or both.</li> <li>File usage.</li> <li>High storage usage. Same subprograms as the affected transaction.</li> <li>Same access method or CICS features as used by transaction.</li> <li>Limit conditions.</li> </ul> | <ul style="list-style-type: none"> <li>As for level 2, increase capacity, reduce data transfer, or change transaction logic or file design.</li> <li>Change file placement buffer allocations or change enqueue logic or file design.</li> <li>Redesign or tune applications, or both.</li> <li>Redesign or tune application subprograms, or both.</li> <li>Reallocate resource or change application.</li> <li>Reevaluate use of feature in question.</li> </ul> |
| 4     | Poor response for specific terminals.         | <ul style="list-style-type: none"> <li>Check network loading as appropriate.</li> <li>Check operator techniques.</li> <li>Check WD entries.</li> </ul>                                                                                                                                                                       | <ul style="list-style-type: none"> <li>Increase capacity of that part of network.</li> <li>Revise terminal procedures.</li> <li>Redefine WD entries.</li> </ul>                                                                                                                                                                                                                                                                                                   |

## Performance considerations for system administrators

This section identifies the performance issues that generally are addressed by system administrators.

### Improving performance by examining startup and shutdown processes

Table 36 describes the startup sequence for CICS (including SFS) and summarizes how startup and shutdown affect performance. This is the full sequence; if any components are already running, the appropriate section does not apply.

Table 36. Performance cost of starting up CICS

| Startup task               | Performance cost                                                                                                                                                                                                                                                                                                                                                                      | Possible action to improve performance                                                                                                                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File system runs recovery. | If the file system terminates abnormally, it redoes all the work since the most recent checkpoint. The file system commits any transactions that were committed before the termination, rolls back any transactions that did not reach syncpoint, and takes locks again on behalf of any transactions that reached the prepare phase of syncpoint but did not reach the commit phase. | You can make the checkpoint interval small, for example, one minute; then the file system has less than one minute's worth of work to redo. However, every minute during normal operation while the file system takes its checkpoints is an overhead cost. |

Table 36. Performance cost of starting up CICS (continued)

| Startup task                   | Performance cost                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Possible action to improve performance            |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| The CICS region runs recovery. | If the region terminated abnormally or was shut down with the immediate option, it recovers the pointers for TS and TD queues from the file system, and it contacts the appropriate file system and other CICS regions to resolve any transactions that were between the prepare and commit phases of syncpoint at the time of the crash. CICS can restart even if it cannot contact these other applications; in this case, records and queues that are locked by affected transactions remain locked until the transactions are resolved. The CICS region then attempts to recover each terminal that was attached at the time of the failure. These are recovered sequentially; if the terminal is available, it is recovered rapidly, but if the terminal is unavailable, it times out after eight seconds. A region that terminates abnormally takes a long time in this phase if many terminals become unavailable before restart. | Always shut down the region normally if possible. |

Redundant CICS regions in your system can adversely affect your system's performance. You can remove such redundant regions by restarting your system. If you decide to do this, it is important that you shut down the components of your CICS system in the following sequence:

1. CICS region
2. SFS server

If CICS does not terminate after five minutes, you can use the operating system **kill** command to force termination.

## Improving DB2 performance by tuning parameters

DB2 provides the user with an extensive suite of tuning parameters. These parameters are flagged as having a performance impact of high, medium, or low to give an indication of the relative importance of each. A detailed description of the parameters is given in the *DB2: Administration Guide*.

The configurable parameters are defined under two categories:

- Database manager categories affect all databases.
- Database parameters affect individual databases.

DB2 also provides a Database System Monitor, which gathers statistical information about the operation of the database manager and activity information



about database processing. You can use this information to decide which configuration parameters to tune. DB2 monitoring is described in the *DATABASE 2 System Monitor Guide and Reference*.

The following parameters have a particular importance for CICS. See the *DB2: Administration Guide* for details.

- **tp\_mon\_name:** This database configuration parameter identifies the name of the transaction processing monitor product that is being used. It is effective when DB2 two-phase commit is used. Set this parameter to **cics**. When set, this parameter enables XA\_END chaining.
- **mincommit:** This parameter allows you to delay the writing of log records to disk until a minimum number of commits have been performed or until a one-second timeout expires.

When appropriate for the user application, and where a number of application servers are connected to a database, the grouping of commits can reduce the overhead of DB2 log record writes and improve performance. The number of applications that are connected to the database must be greater than or equal to this value.

- **num\_ioservers:** This parameter defines the number of I/O servers that are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O.
- **maxagents:** This parameter defines the maximum number of database manager agents that are available at any given time to accept application requests.
- **maxcagents:** This parameter defines the maximum number of database manager agents that can be concurrently executing a database manager transaction.

In general, where applications make heavy use of writing to TD and TS queues, a performance improvement can be expected if DMS (Database Managed Space) table space is used instead of the default SMS (System Managed Space) table space.

With DMS table space, the database manager controls the storage space instead of the operating system's file system manager as is the case with SMS table space.

See the *DB2: Administration Guide* for details about how to create DMS table space.

## Improving performance by adjusting settings for file control

You can improve the performance of file control by adjusting the following settings:

- Physical file placement.

Minimize the number of *hops* between the transaction program and its files. In particular, if a file is managed by an SFS server anywhere in the network, configure CICS to access it directly instead of by function shipping through another CICS region.

- Recoverability of files.

Some files do not need a recoverability guarantee; for example, an audit trail file can possibly tolerate additional records being written even if the enclosing transaction subsequently rolls back. Defining such files as nonrecoverable allows syncpoints to be run more cheaply. Also, files that are read-only can be defined as nonrecoverable and can be backed up elsewhere in case of a system or disk failure.

- SFS buffer size.



Choose a value that is applicable to the size of your file system and hardware. The value is set in the Structured File Server Definitions (SSD). The default is 1 MB; it can be increased to 200 MB.

- Error isolation.

The default value for the File Definitions (FD) **ErrorIsolation** attribute is **on**, in which case, all SFS errors are passed on to the application. Performance improves slightly if you set this attribute to **off** because the time that is taken to pass on and handle SFS errors is eliminated.

- Using a browse cache.

The CICS\_BROWSE\_CACHE environment variable is used to specify a cache size that is to be used for browsing files on an SFS server. The cache is set up in multiples of 4-KB records. Consider carefully whether to use a browse cache and if you use one, how large a cache to use. Keep in mind that all file browse operations use the cache. In some cases its use improves performance, but performance can be degraded if multiple short browses are used.

## Improving performance by uninstalling inactive clients

The RD attribute **ISCDelayMinutes** specifies the time period between instances of the CICS private transaction CFTS, which is used to delete shipped terminal definitions. The CFTS transaction also informs remote systems that transaction-routed terminals have been uninstalled, enabling the resources to be freed.

The default value of the **ISCDelayMinutes** attribute is 10. Under normal conditions, you can set this attribute to poll for inactive clients every 30 minutes. The interval can be shortened if you suspect the client of being inactive. If the **ISCDelayMinutes** attribute is set to a value such that inactive clients are not polled, system resources are not freed, and the region's performance is reduced.

## Improving performance by controlling the rate of requests to the RPC listener process

External requests to install terminals and to execute transactions can arrive in the region from several sources. The CICS RPC listener process (**cicsrl**) takes requests for **cicslterm**, **cicsteld**, and PPC Gateway servers. The requests are validated and can then be executed in an application server process.

The rate at which these requests are processed by the RPC listener process is governed by the number of RPC listener threads that are specified by the process during CICS initialization. You can explicitly specify this value by setting the Region Definitions (RD) attribute **RPCListenerThreads** to the desired value. Alternatively, you can leave the **RPCListenerThreads** attribute set to the default value of 0 and let CICS calculate the number of RPC listener threads to be allocated. By default, the RD attribute **RPCListenerThreads** is set to 0 and the RPC listener process calculates the number of RPC listener threads to be ten times the value of the RD **MaxServer** attribute.

However, each operating system has a maximum number of threads that can be concurrently supported. If the default calculation of ten times the value of the **MaxServer** attribute results in a number that exceeds the supported limit of concurrent threads, ensure that the value for the **RPCListenerThreads** attribute is explicitly set to an acceptable value. See Table 37 on page 234 for information about these limits.

Table 37. CICS response time checklist

| Operating System | System limit for concurrent threads                                                                          |
|------------------|--------------------------------------------------------------------------------------------------------------|
| AIX              | 32768                                                                                                        |
| HP-UX            | Determined by settings of kernel parameters. See the <i>TXSeries for Multiplatforms Installation Guide</i> . |
| Solaris          | $(2^{**}32) - 1$                                                                                             |

**On the AIX platform.** The RD attribute **RPCListenerThreads** controls the number of threads that are available for incoming RPCs from CICS on Open Systems clients, IBM CICS Universal Clients, and PPC requests. The value for this attribute is calculated to be ten times the value of the RD **MaxServer** attribute. AIX 4.3.2 has a limit of 512 threads per process. If the value of the **MaxServer** attribute is greater than **49**, the **RPCListenerThreads** attribute must be set to a value that is less than **499**. Instead of allocating the maximum number of threads, specify a lower value and monitor the RPC statistics to see whether the number of available threads becomes exhausted. (Note that each idle RPC thread uses a kernel thread, which can use system resources unnecessarily.) However, on AIX 4.3, it is possible to compile by using the `-D_LARGE_THREADS` compiler directive to enable 32767 threads per process.

Each RPC listener thread that is actively validating a request has an associated RPC queue that can hold up to eight further requests. Requests that are submitted to the RPC listener process are validated immediately if a free RPC listener thread is available; otherwise, the request waits on an RPC queue until an RPC listener thread becomes available. When all RPC listener threads are busy and all associated RPC queues are full, any further requests are rejected by the RPC listener process.

Conversational, long-running pseudoconversational transactions, and autoinstall requests can cause delays when no RPC listener threads are available. When approximately 90 percent of the RPC listener threads are busy or waiting for requests to complete, CICS manages further requests by processing. Each further autoinstall request still occupies one of these RPC listener threads until the autoinstall is completed.

## Improving performance by controlling the number of transactions running in a region

You can control the number of transactions that are running in a region at any one time by using *transaction classes*. You can assign a transaction to a class by specifying the **TClass** attribute in the Transaction Definitions (TD). You can use the **ClassMaxTasks** attribute in the Region Definitions (RD) to limit the number of transactions that can be run simultaneously in a given class.

CICS permits 11 transaction classes: classes **1** through **10** and **none**. Only classes **1** through **10** can be controlled by using **ClassMaxTasks**. Transactions that are defined with class **none** are not subject to any class restriction. All CICS-supplied transactions default to class **none**.

To use transaction classes to ensure that CEMT is run at any time:

- Assign all user transactions to a class other than **none** (including the CICS-supplied user transactions).
- Leave CEMT defined with class **none**.

- Set **MaxServer** to be at least one greater than the sum of the values that is specified in **ClassMaxTasks**.

The number of transactions that can run in a region at any given time is controlled by the **ClassMaxTasks** and **MaxServer** attributes in the Region Definitions (RD).

## Improving performance by tuning I/O access

You can use operating system facilities such as the **vmstat** command to determine whether a busy disk is constraining the throughput of the system.

(On the AIX platform. You can also use the **sar** command for this task.)

If you see a significant I/O wait in a system that can be doing work for other users, that system is being constrained in this way.

The possible actions are:

- Install more physical memory.
- Tune the placement of data on the disks.
- Run more application servers.
- Install faster disks.
- Configure mirrored volumes.
- Configure striped volumes.

The following indications can be investigated:

- The **vmstat** command shows large numbers of processes that are blocked for page-in (in column b).

This suggests high paging or swapping activity. Reduce the number of application servers or install more physical memory. Do not take any more tuning actions until this has been done.

- CICS does not initiate enough concurrent transactions.

Each conversational transaction occupies an application server for the duration of the transaction. The remaining application servers represent the largest number of concurrent transactions that can be initiated. If this number is small, the system does not initiate enough work to overlap disk wait. In particular, the total number of application servers that are available for pseudoconversational work in a network of CICS regions should not be less than the number of SFS servers that are managing files on that network.

- The **filemon** command shows usage rates of logical and physical volumes.

Statistics from the operating system **filemon** command indicate the average data-transfer rate and the seek pattern for each physical volume and the usage pattern for each logical volume.

Volumes that are used as logs are written to during every syncpoint. It can be desirable to put each of these on its own physical volume. Otherwise, a large amount of seeking can occur to get between the database and the log area.

Adding physical memory decreases the need for disk usage for purposes other than syncpoint.

Mirrored volumes usually schedule as many concurrent independent reads as there are mirror copies (two or three); this can increase the time for a write because the subsystem must ensure that all copies are updated.

**On the AIX platform.** When you allocate a logical volume, you can choose whether it is spread over a maximum or minimum number of physical volumes. Specifying the maximum for a striped volume spreads the data over several physical volumes, which improves performance but increases the risk of losing some data.

If you allow the AIX operating system to place a logical volume across more than one disk, it stripes automatically.

**On the HP-UX or the Solaris platforms.** Striped volumes improve write performance by writing to a particular file system across multiple disks.

## Improving performance by balancing throughput and response time

You can configure CICS to run with a minimum response time or with a maximum throughput.

The following specifications affect this decision:

- **MaxServer** attribute in the CICS Region Definitions (RD): Specifies the maximum number of tasks in the system at any one time.  
If this is too small, processing time can be wasted because not enough work is in the system for disk read/write operations to be overlapped with other transactions.  
If this is too large, the system can spend more time managing the servers than running the transactions.
- **OpThreadPoolSize** attribute in the Structured File Server Definitions (SSD) for the SFS server: Determines the maximum number of simultaneous file control operations that can be in progress in an SFS server at a time. Additional requests are queued at entry to the SFS server. A value of **10** is usually appropriate.

## Improving performance by using an SP2 on the AIX platform

**On the AIX platform.** You can significantly improve the performance of your CICS system with the IBM RS/6000 Scalable POWERparallel System (SP2). An SP2 system contains several processor nodes and can be set up to use a High Performance Switch (HPS) for internode communications. The *SP2 Administration Guide* describes how to configure and start the HPS.

---

## Performance considerations shared by administrators and developers

This section identifies performance issues that can require consideration from both the system administrator and the developer.

## Improving performance by careful use of CICS facilities

The following CICS facilities can affect performance.

- CICS monitoring and trace.  
You can turn off trace and monitoring when you do not need them. You can also tailor the trace and monitoring facilities so that only a minimum amount of statistics is produced.
- Using CEDF.  
To avoid the performance costs of this type of debugging, do not use the **-e** option when compiling with **cicstran**.
- Using main instead of auxiliary temporary storage.

If application logic allows, use main temporary storage instead of auxiliary temporary storage to maximize performance.

- Using multiple regions.
- Recoverable resources.

If you use files, transient data queues, and temporary storage queues that are read-only, they can be defined as nonrecoverable and can be backed up elsewhere in case of a system or disk failure.

- Recoverable files.

Some files do not need this guarantee; for example, an audit trail file can possibly tolerate additional records being written even if the enclosing transaction subsequently rolls back. Defining such files as nonrecoverable allows syncpoints to be run more cheaply in some cases. Defining read-only files as nonrecoverable gives better performance than does defining them as recoverable.

Note that if you are using DB2 for queue and file management, all nonrecoverable files are treated as recoverable.

- Syncpoints.

For best syncpoint performance, minimize the number of participants in a transaction, particularly the participants that are updated.

- Region hash table size.

In general, set hash tables to twice the number of entries for each resource in the Region Definitions (RD).

- CICS application servers.

Make sure you set the RD **MaxServer**, **MinServer**, and **ServerIdleLimit** attributes to appropriate values.

- CICS internal checking.

## Distributing the workload between regions

You can distribute the workload in a multiregion environment to improve the system's response time, capacity, and availability as perceived by users of CICS transactions. Route transactions to other CICS regions or use the dynamic transaction user exit.

### Routing transactions to other CICS regions

When you define a transaction to CICS, you can determine whether the region is to handle it as a *local* or a *remote* transaction. By default, transactions are defined as local, which means that they always run in the CICS region in which they are started. You can, however, define a transaction to be remote, in which case it is routed to another CICS region to be executed. To define a transaction as remote, include a specification of one or both of the following attributes in the Transaction Definitions (TD):

- **RemoteSysId**: The name of the remote CICS system or region where the transaction resides.
- **RemoteName**: The name by which the transaction is known on the remote system, if it is different from the name that is defined in your local region.

### The dynamic transaction routing user exit

It can be helpful to have more than one region to which a transaction can be routed. This can be helpful for many conditions; for example, when the transaction is routed to a region that fails to start. By setting the **Dynamic** attribute in the Transaction Definitions (TD) to **yes**, you can have the local execution or routing of the transaction handled by the dynamic transaction routing user exit. See "Working with CICS user exits" on page 185 for more information.

---

## Possible effects of changing the system clock

Some CICS products store their own system clock, which is set at CICS startup. With these CICS products, you must restart CICS to change the system clock. This does not, however, apply to CICS on Open Systems.

CICS always looks at the operating system clock to obtain the time. If you update the operating system clock, CICS picks up the updated time immediately, provided that the appropriate thread is active. If the appropriate thread is not active, CICS does not know that the operating system clock has been updated. To ensure that the thread is active, you can run an EXEC CICS START command.

You can write a transaction to issue an EXEC CICS START NOW command and do something harmless. Each time you change the system clock, you can run this transaction to ensure that CICS recognizes the new time immediately.

Local time zones, time changes from winter to summer, and the date of the change, are handled by the environment variable that specifies the time zone for the operating system. This is set before the region is started and enables the region to behave correctly across winter and summer time changes without requiring a restart.

If the operating system resets the clock because of a seasonal time changeover, the intervals and schedules that are specified for CICS transactions can be affected as follows:

- Changing from Winter Time (Standard Time) to Summer Time (Daylight Savings Time):

At 18:00, a transaction issues an EXEC CICS START command with a time interval, so that the started transaction is to run at 04:00. At 02:00, the operating system changes the system clock to 03:00. The started transaction runs at 05:00 instead of 04:00.

- Changing from Summer Time (Daylight Savings Time) to Winter Time (Standard Time):

At 18:00, a transaction issues an EXEC CICS START command with a time interval, so that the started transaction is to run at 04:00. At 02:00, operating system changes the system clock to 01:00. The started transaction runs at 03:00 instead of 04:00.

---

## Chapter 20. Understanding CICS backup and recovery

This chapter describes ways in which you can protect your CICS region's configuration and business data against loss or damage from uncontrolled system shutdowns. It also describes some of the mechanisms that CICS provides to help you do this.

See related information in "Recovery and restart for CICS startup" on page 200, "File manager unavailable at startup and while running transactions" on page 201, "CICS shutdown processes" on page 207, the *DB2: Administration Guide*, the *DB2: Programming Guide*, and the *TXSeries for Multiplatforms Intercommunication Guide*.

"Protecting your CICS system and data," "Preparing your CICS systems for recovery" on page 241, "Suggestions for backing up your systems" on page 252 and "Improving your ability to restore services" on page 253 discuss the details that you must consider to ensure your ability to restore your CICS system if a system failure occurs.

---

### Protecting your CICS system and data

To protect your system and data against loss or damage from uncontrolled shutdowns, and to enable you to restore services quickly, you need to implement a backup and recovery strategy.

Typically, a backup and recovery strategy requires the following:

- Defining as recoverable those resources that contain important data that can be changed by CICS transactions.
- Backing up the resource definitions in the permanent resource definition database.
- Ensuring that applications that can update data in the region are programmed for recovery. (This is described in the *TXSeries for Multiplatforms Application Programming Guide*).
- Backing up the application data.
- Backing up the logs of transactional updates.

---

### Mechanisms that aid backup, recovery, and restart

CICS and the file manager provide the following mechanisms that you can use in your backup and recovery strategy:

- Logical units of work (LUWs)
- Recoverable resources
- Dynamic transaction backout
- The CICS log
- User journals
- Automatic start
- Transaction logging
- Forward recovery

### Synchronization points (syncpoints)

The end of a logical unit of work is indicated to CICS by a *synchronization point*, usually abbreviated to *syncpoint*.



A syncpoint can be caused in the following ways:

- Implicitly at the end of a transaction, signaled by an EXEC CICS RETURN command at the highest logical level. This means that a logical unit of work cannot span tasks.
- Explicitly by EXEC CICS SYNCPOINT commands that are issued by the application program at logically appropriate points in the transaction.

It follows from this that an LUW starts:

- At the beginning of a task
- Whenever a transaction issues an explicit EXEC CICS SYNCPOINT command

An LUW that does not change a recoverable resource has no meaningful effect, either for you or for the CICS recovery mechanisms. CICS does not commit resource changes by a task until CICS reaches a synchronization point. If CICS interrupts task processing because of an error of any kind, CICS backs out changes made within the abnormally terminating LUW.

## Two-phase commit

To provide the ACID properties, CICS implements a protocol known as *two-phase commit*. When an LUW is to be committed, CICS needs to ensure that all recoverable servers that are involved in the LUW commit their updates; if one or more cannot do so (for example, the communications line to the server has broken), all servers must back out their updates. To do this, the commit procedure has two phases. In the first phase, CICS asks each participant in turn to record enough information about the LUW to enable CICS to commit or back out the updates in the second phase. This is called *preparing* the LUW. Each participant can respond to CICS in one of three ways:

- yes: The participant is to commit or back out, as instructed.
- no: The participant wants the LUW to be backed out.
- read only: The participant does not care about this LUW.

If a participant responds with yes, but commit or backout has not yet occurred, the participant has the status `indoubt`.

If none of the responses is no, CICS instructs all participants to commit the LUW. If any response is no, CICS instead instructs all participants to back out. This is the second phase of the two-phase procedure.

## CICS recoverable resources

To ensure that your business data is updated in accordance with the ACID principles, define the resources in which the data is stored as *recoverable* resources.

The following resources can be defined as recoverable:

- Data files
- Intrapartition transient data (logically recoverable)
- Auxiliary temporary storage queues
- Autoinstalled terminal definitions
- EXEC CICS START PROTECT REQUESTS

See the *TXSeries for Multiplatforms Administration Reference* for information about the relevant resource definitions:

- The description of the **RecoveryStatus** attribute in "File Definitions (FD)"

- The description of the **RecoveryType** attribute in "Transient Data Definitions (TDD)"
- The description of the **RecoverFlag** attribute in "Temporary Storage Definitions (TSD)"

During normal operation and shutdown, CICS and the recoverable servers with which it communicates record information that is required for restart. If a complete or partial error or a shutdown occurs, this information enables CICS to recover and restart the region. During automatic restart, CICS returns recoverable resources to their committed states.

If a transaction terminates abnormally and the resources that are affected by the terminating task are recoverable, CICS automatically performs dynamic transaction backout. However, after dynamic transaction backout, CICS does not automatically restart the transaction. Unless the file is on a DB2 database, where all files are recoverable, you *must* define a resource as recoverable in order for CICS to back out changes.

## Automatic start

You can use the **StartType** attribute setting **auto** to start the SFS server after a failure. The CICS log is required for a successful restart of the SFS server. If the CICS log is not available, a cold start of the SFS server is required.

---

## Preparing your CICS systems for recovery

The information in this section describes how to enable your CICS system for recovery if a failure occurs that causes the data on the CICS system to be unavailable.

Recovering data after an individual transaction fails, and even after some unexpected system shutdowns, can be relatively easy, as long as you have defined resources as recoverable. CICS and the SFS server recover the data in such resources automatically when you restart the system. All transactions that are committed before the shutdown are still committed, and all transactions not yet committed are fully rolled back.

Recovering data that has been lost from a disk, however, can be more complex and requires planning and implementation of procedures to cope with such a failure. The procedures that enable recovery and restart of a CICS system in the event of these types of failures depend on the components of that system, the type of work that is performed on it, and the degree of recoverability required. Each individual component of the CICS system must be considered, in addition to the system as a whole.

Recovery of your SFS server and your application data are discussed in "Enabling recovery of the SFS server" on page 242 and "Backing up the SFS logical volumes" on page 242. DB2 recovery techniques are described in "Enabling recovery of DB2" on page 247. Oracle recovery techniques are described in "Enabling recovery of Oracle" on page 249.

Information about recovering your CICS region, CICS subsystems, and CICS user IDs is provided in "Enabling recovery of CICS" on page 249, "Enabling recovery of the CICS subsystem entries" on page 251, and "Enabling recovery of the CICS user IDs" on page 251.

Full recovery of a CICS system following a system failure is discussed in “Full recovery after a complete system failure” on page 251.

## Enabling recovery of the SFS server

Configuration and running information for all SFS servers that are on a machine is located in the `/var/cics_servers` directory. This directory contains the following subdirectories:

- The GSD directory, which contains the Gateway Server Definitions (GSD) autostart and cold start databases for any PPC Gateway server that exists on the machine, and information that is unique to each gateway.
- The SCD directory, which contains the Schema File Definitions (SCD) autostart and cold start databases for all SFS servers that are on the machine.
- The SSD directory, which contains the Structured File Server Definitions (SSD) autostart and cold start databases for all SFS servers that are on the machine, and information that is unique to each SFS server. For each SFS server, the SSD directory contains the following files:
  - A *lock file* that is used to lock a file while it is being updated. The full pathname for the lock file is `/var/cics_servers/SSD/cics/sfs/baseName/lock`.
  - A *message file* that contains the messages that are logged for the server. The full pathname for the messages file is `/var/cics_servers/SSD/cics/sfs/baseName/msg`.
  - Two *restart files* that contain information that the server needs for a warm start. The full pathnames for the restart files are `/var/cics_servers/SSD/cics/sfs/baseName/restart` and `/var/cics_servers/SSD/cics/sfs/baseName/restart.bak`.

Restoring a CICS system, which is discussed in “Recovering SFS queues and file management” on page 246, involves the following procedures:

- Installing CICS. This creates the SFS directory structure that is common to every SFS server.
- Re-creating and reconfiguring the SFS server. This creates the directory structure for the SFS server, including the lock file, the message file, and the restart files.
- Recovering application data, if necessary, by restoring the SFS logical volumes from backup or from the mirrored SFS logical volumes.

Media recovery archiving (MRA) files can be used to restore application data up to and including the data that is on the last available MRA file. This can possibly be a system-available MRA file that has not yet been backed up. MRA is described in “Backing up the SFS server by using MRA” on page 243.

**Note:** Because MRA does not back up the CICS log, which is located in the `/var/cics_regions/regionName` directory, MRA is not useful if you need to restore the application data in a consistent level with the region. The reason for this is that the CICS log must be at the same processing point as the SFS logical volumes for the data to be meaningful. You must take backups when the region and file manager are shut down, to ensure that copies of the CICS log and the data are at the same point in processing.

## Backing up the SFS logical volumes

The SFS server manages the data that is associated with CICS files and queues on two logical volumes, one for queue and file data and one for log data. When you cold start the SFS server, the data on these logical volumes is discarded.

**Online backups using data mirroring:** The best way of ensuring that the SFS data is recoverable is to use online data mirroring for the file system that contains the logical volumes.

**Choosing an offline backup method for the SFS server:** Two options are available for offline backups of the SFS server data:

- Media Recovery Archiving (MRA), which is provided by CICS SFS
- Copying the data by using operating system facilities

The use of MRA or the operating system for backing up the SFS server has advantages and disadvantages. When deciding which to use, consider the following points:

- MRA enables backups to be taken while the SFS server is running. Because the log archive files are continuously written, you do not need to take a frequent backup of the SFS data. The data can always be restored to the level that is consistent with the latest log archive file.
- Copying the logical volumes has the disadvantage that the SFS server must be stopped when the backup is taken and the data can be restored only to its level when that backup was taken. This means that you need to take more frequent backups.
- The continuous writing of archive files by MRA can also be considered a disadvantage. It causes some degradation in the performance of the SFS server and the files require disk space. If not correctly maintained, the files that are written by MRA can cause the SFS server to terminate. To avoid problems with the SFS server, copy the archive files regularly to ensure that the file system on which they reside does not fill up.
- Copying the logical volumes has the advantage that the data is restored to a known level (its level when the copy was taken). As a result, if the CICS log was saved at the same time, the region can be restarted to a known level.
- With MRA, if the log volume is lost, the level to which the data volume can be restored is not fully defined, although it will probably be more recent than the backup of the data volume. This means that the data cannot be restored to a level that is consistent with a saved CICS log.
- File activity is initially logged by CICS SFS, and is flushed to an MRA log file only when a particular amount of file activity has taken place. Because of this, if you take an MRA backup at a specified time, you have no guarantee that *all* file activity is included in the backup.
- MRA has the advantage that, if only the data volume is lost, it can be restored from backup to the level at which it was when it was lost. In this respect only, it can be considered as an alternative to mirroring the SFS data volume.

**Backing up the SFS server by using MRA:** Media Recovery Archiving (MRA) is a function that CICS SFS provides to support the backup and recovery of SFS servers. It enables you to create an offline backup of the SFS server with no disruption in service. The procedure is defined as follows:

1. The server is running with MRA enabled. See the “Enabling and disabling media archiving” on page 245 for information about how to do this.
2. A full backup of the SFS data file logical volume is created and is either saved to tape or stored on another system.

**Note:** This backup can be created in increments (fuzzy backup); however, restoration is not possible until a full backup has been taken.

3. The restart files, **restart** and **restart.bak**, are saved from the home directory of the server to tape or to another system.

**Note:** When these restart files have been saved once, they must be saved again only when the logical volumes that the server owns are changed.

4. The SFS server continues to be used.
5. Each time the SFS log file logical volume is filled to a specified level, SFS dumps the state of the log file logical volume to an archive file.
6. The log archive files are saved periodically.

This procedure must be followed at regular intervals.

If a system failure occurs and you restore the restart files, the recovery procedure depends on whether the log file logical volume has been lost.

If the log file logical volume is still intact, the SFS data file logical volume can be restored from backup. This process replays the log records that have been written since the backup was taken. This action restores the data volume to the state it was in immediately before the data volume was lost. However, it requires all log archive files that were written since the data backup to be available.

If the log file logical volume is lost, irrespective of whether the data volume is lost, you must first restore the log file logical volume and then the data volumes from the archive files.

You must restore the log file logical volume from the latest available log archive file. This process can also require older log archive files, and restores the log file logical volume to the state it was in when the latest available log archive file was written. The data volume is then restored in the same way as before. That is, it replays the log records that have been written since the backup was taken. This action restores the data volume to the state it was in when the latest available log archive file was written. This, almost certainly, involves some loss of data.

If you use MRA, ensure that you reduce the number of log archive files on the system periodically. These files can be very large, and if they are left to fill up the volume, the SFS server cannot archive any data, and terminates abnormally. At appropriate intervals, save these files and delete them from the directory. For an SFS named `./cics/sfs/hostB`, the files are written to the `/var/cics_servers/archives/hostB` directory.

Alternatively, you can use the SFS Diagnostic Tool (SDT) to convert your SFS files into flat files, which you can then store on another system. For information about using the SDT, see the *TXSeries for Multiplatforms Administration Reference*.

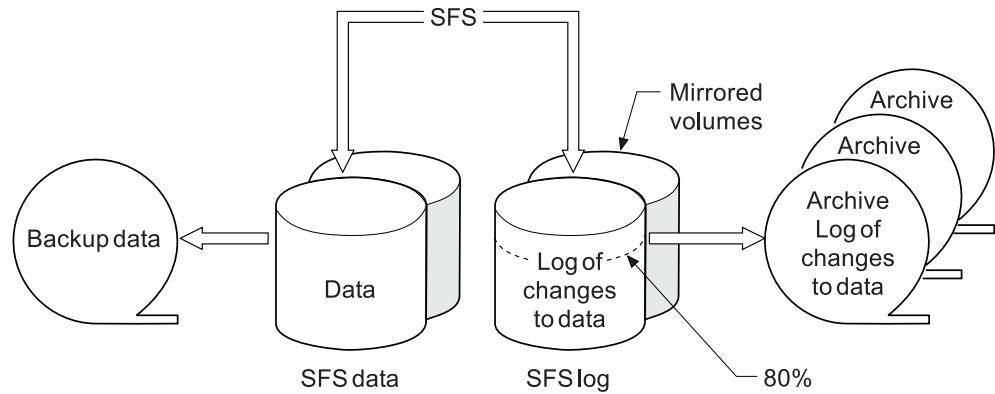


Figure 11. Media Recovery Archiving (MRA) used in combination with mirroring for SFS backup

**Using the operating system to copy the SFS logical volumes:** You can use the operating system to copy the whole contents of a logical volume. For an SFS server, the restart files, the data volume, and the log volume must be copied in a consistent state. The only way you can do this is to copy them simultaneously while the SFS server is not running. The restart files (**restart** and **restart.bak**) are located in the home directory of the server.

If a system failure occurs, you can restore the SFS data and log volumes from the latest level available, then autostart the SFS server.

**Enabling and disabling media archiving:** Media archiving is enabled and disabled by setting the **MRAArchivingEnabled** attribute in the SSD. By default, this attribute is set to **no**. Use the following procedure to set the value of the attribute:

1. Shut down the SFS server.
2. Use the **cicsupdate** command to change the **MRAArchivingEnabled** SSD attribute to either **yes** or **no**. For more information about this definition, see *TXSeries for Multiplatforms Administration Reference*. See "Changing a resource definition in the permanent, autostart, and runtime databases" on page 117.
 

```
cicsupdateclass -w -c ssd MRAArchivingEnabled -n yes
cicsupdateclass -w -c ssd MRAArchivingEnabled -n no
```
3. Warm start the SFS server.

Alternatively, you can enable or disable media archiving without stopping the SFS server by using the **tkadmin** command to override the SSD **MRAArchivingEnabled** attribute. The **tkadmin** command affects the setting for media archiving only for the current SFS run and does not permanently change the SSD attribute.

```
tkadmin enable mediaarchiving -server serverId
tkadmin disable mediaarchiving -server serverId
```

Also, the **tkadmin** command can be used to query whether media archiving is currently enabled or disabled. The following example shows the command syntax:

```
tkadmin query mediaarchiving -server serverId
```

**Establishing a baseline backup:** To establish a baseline for recovery when using media archiving, use the CICS SFS recovery and log services to back up this SFS server's logical volume or data volumes. Because the backup files are created by the SFS server, the SFS server must have operating system read and write



permissions for the directory in which you intend to create the backup segments. An example command sequence for this is:

```
mkdir backup-directory
chown SFS2 backup-directory
chmod 777 backup-directory
cd backup-directory
tkadmin backup lvol -server /./cics/sfs/hostABCD sfs_ShostABC
```

*Recommendations about media archiving backup:* To restore SFS data, the backup path and the SFS backup filenames are required. This information is logged in a media archiving file when a particular threshold of data activity has occurred. Therefore, this media archive file holds the required SFS backup information that is needed to restore the log data.

When log data is destroyed and few media archiving backup files are available after an SFS data backup, a remote possibility exists that the required SFS backup information has not been logged to a media archiving file. In that case, the SFS data cannot be restored. To avoid this condition, consider taking the following precautions:

- Perform the SFS fuzzy backup in partial stages; this is the process for which SFS backup was principally designed.
- Minimize the risk of losing data by moving all backups offline as soon as they are completed.

See *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration* for detailed information about mirroring, media archiving, and backup and recovery procedures.

## Recovering SFS queues and file management

This information describes how to recover an SFS server by moving it to a new machine. It assumes that you have backups of the SFS data file logical volume, the SFS log file logical volume (see “Enabling recovery of the SFS server” on page 242).

Use the following steps to recover the SFS server:

1. Install CICS onto the new machine.
2. Ensure that the machine has the same TCP/IP name and address as the machine does from which the backups were made.
3. Create the SFS server to match the original SFS server. See “Administration of file and queue management on an SFS server” on page 23.  
The Structured File Server Definitions (SSD) for this SFS must exactly match the original SSD of the SFS server that you are recovering.
4. Cold start the SFS server. See “Defining an SFS server by using the `cicssfscscreate` command” on page 275.
5. Shut down the SFS server.
6. Recover the SFS log file logical volume. The method that you use depends on the backups that are available.

If the SFS server was running with media recovery archiving (MRA) enabled, the log volume can be recovered by using the log archive files. To do this, you must start the SFS server with no log volumes. Use the `cicssfscs` command with the `-nlv` option. Then use the `tkadmin restore logvols` command. When this method is used, the log volume is recovered to the state that it was in when the



last log archive file was written. For information about using these commands, see the *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration* .

If an operating system command was used to back up the log file logical volume when the SFS server was stopped, the copy can be used to recover the logical volume to the state that it was in when the copy was taken. For this type of recovery, the SFS server must be idle.

7. Recover the SFS data file logical volume.

If the SFS log file logical volume and the SFS data file logical volumes must be recovered, recover the SFS data file logical volume only after the log file logical volume has been recovered. The method that you use depends on which backups are available and how the log file logical volume was recovered.

If the **tkadmin backup lvol** command was used to take a backup of the SFS data file logical volume, the data can be recovered by using the **tkadmin recover lvols** command. This recovers the data to a state that is consistent with the recovered state of the log file logical volume. To do this, the SFS server must remain in no log volume mode (which is started by using the **-nlv** option of the **cicssfs** command) as required for the recovery of the SFS log using media recovery archiving.

If the backup was taken by using an operating system command, the copy of the backup can be used to recover the data file logical volume to the state that it was in when the copy was made. However, for this to be a successful recovery, the log file logical volume must be recovered to exactly the same level at which the data file logical volume is. In practice, this means that copies of both logical volumes must be taken at the same time when the SFS server is idle.

8. When both logical volumes are recovered, autostart the SFS server in normal mode rather than in no log volume mode.

You can test that data has been recovered by showing a list of files on the SFS server. For example, if the SFS server ID is **hostB**, use the following command:

```
sfsadmin list files -server ./:/cics/sfs/hostB
```

You can also define and start a CICS region and use it to issue transactions to test the validity of the SFS data.

Table 38 shows the location of the example files for each supported platform:

*Table 38. Location of example files for testing the validity of SFS data*

| Platform | Directory                                       |
|----------|-------------------------------------------------|
| AIX      | /usr/lpp/cics/src/examples/backup/backup_sfs.sh |
| HP-UX    | /opt/cics/src/examples/backup/backup_sfs.sh     |
| Solaris  | /opt/cics/src/examples/backup/backup_sfs.sh     |

For detailed information about backing up and restoring SFS data and log files, see the *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

## Enabling recovery of DB2

Two types of recovery are available:

- Restoring to a previous backup (*version recovery*)
- Restoring to the point of failure

## Enabling version recovery

To be able to restore the data to a previous backup, you need a backup of the database that was taken at the same time that a backup of the CICS log was taken. The reason for this is that the data and the CICS log need to be restored to the same processing point.

Use the DB2 BACKUP DATABASE utility to take offline backups of the DB2 data. Use operating system facilities to take offline backups of the CICS log. Both these backups must be taken when the region is shut down and the database is offline. Otherwise, the CICS log and the database can be unsynchronized, and the ability to recover the data will be unreliable.

## Enabling point of failure recovery

To be able to restore the data and applications to the point of failure, you need a copy of both the database and the CICS log that was taken when the failure occurred. To do this:

- You need to mirror the CICS log. The operating system documentation describes how to use mirroring to create an offline image backup of an active file system.
- You need to either mirror the logical volumes that contain the DB2 data, or enable roll-forward recovery. The operating system documentation describes how to mirror logical volumes. To use roll-forward recovery, you need the following:
  - An offline backup copy of the database
  - Roll-forward recovery enabled

As described in the *DB2: Administration Guide*, roll-forward recovery allows you to build on a restored backup to a specified point in time. The database must be configured to enable roll-forward recovery. This is done by setting specified parameters in the DB2 database configuration file. For example, the **logprimary** parameter is used to set the number of primary log files that are created, and the **logfilsiz** parameter is used to set the number of pages for each log file.

It is recommended that you mirror the DB2 database, but a lack of coordination is possible between DB2 logging and CICS logging.

## DB2 queue and file management recovery procedure

This information describes how to restore a DB2 queue and file management database. It assumes that you have backups of the DB2 data.

Two types of methods are used to recover a DB2 database:

- *Restore recovery*, which rebuilds a database to the state that it was in when the backup copy was made. This is also referred to in DB2 as *version recovery*.
- *Roll-forward recovery*, which builds on the restored copy of the database, making changes to that copy by using the logs that contain the modifications that have been made since the last backup.

To recover a DB2 database that is used for CICS queue and file management, do the following:

1. Determine whether you will restore the data to a previous backup (*restore recovery*) or to the point at which the failure occurred (*roll-forward recovery*).
2. If you needed to re-create the database, verify that the database is configured for CICS queue and file management as described in “Administration of file and queue management on DB2” on page 31.
3. Start the database.

4. See the *DB2: Administration Guide* for instructions about how to use the RESTORE DATABASE utility to restore the database.

**Note:** The DB2 RESTORE DATABASE utility can be used to create a new database and restore the backup image of the data.

## Enabling recovery of Oracle

Two types of recovery are available:

- Restoring to a previous backup (*version recovery*)
- Restoring to the point of failure

### Enabling version recovery

To be able to restore the data to a previous backup, you need a backup of the database that was taken at the same time that a backup of the CICS log was taken. The reason for this is that the data and the CICS log must be restored to the same processing point.

Use the Oracle utility for database backup to take offline backups of the Oracle data. Use operating system facilities to take offline backups of the CICS log. Both these backups must be taken when the region is shut down and the database is offline. Otherwise, the CICS log and the database can be unsynchronized, and the ability to recover the data will be unreliable.

### Enabling point of failure recovery

To be able to restore the data and applications to the point of failure, you need a copy of the database and of the CICS log that were taken when the failure occurred. To do this:

- You need to mirror the CICS log. The operating system documentation describes how to use mirroring to create an offline image backup of an active file system.
- You need to either mirror the logical volumes that contain the Oracle data or enable roll-forward recovery. The operating system documentation describes how to mirror logical volumes. To use roll-forward recovery, you need the following:
  - An offline backup copy of the database
  - Roll-forward recovery enabled

As described in the Oracle documentation, roll-forward recovery allows you to build on a restored backup to a specified point in time. The database must be configured to enable roll-forward recovery. See the Oracle documentation for information about how to enable roll-forward recovery.

### Oracle queue and file management recovery procedure

You can use the Oracle restore utilities to do a recovery to a particular version or to do a point-in-time recovery. See the Oracle documentation for more details.

## Enabling recovery of CICS

The CICS directory structure for each region is located in `/var/cics_regions/regionName/`, where *regionName* is the name of the region. This directory structure contains the following:

- The permanent database, which contains all resource definitions that are used when the region is cold started
- The runtime database, which contains all resource definitions that are used when the region is autostarted

**Note:** These databases do not contain the Structured File Server Definitions (SSD), Schema File Definitions (SCD), and Gateway Server Definitions (GSD). These are located on `/var/cics_servers/`, as described in “Enabling recovery of the SFS server” on page 242.

- The CICS log, which contains internal information about queues and START requests
- The region environment variables
- Region restart information that is used for warm starts
- The executable files that make up CICS application programs (by default)
- The CSMT transient data queues (TDQ) and any output to statistics files
- Transaction or system dump files
- Basic Mapping Support (BMS) map definitions that are used by CICS application programs (by default)

The information that is contained in the region directory can be restored from backup. However, if you want to restore the application data to the point of failure, you need a copy of the CICS log (in addition to a copy of the SFS logical volumes) that was taken at the point of failure.

## Backing up the CICS log

Whenever a transaction is started in a CICS region, CICS stores information about it in the CICS log. The best way of ensuring that the log data is recoverable is to use data mirroring for the file system that contains the log data. In practice, this means that the logical volume that hosts the file system must be configured for mirroring. It is best to have a dedicated file system for the log data. This gives you greater control over the allocation of file system storage and ensures that this storage can be used only for storing CICS log data.

### Note:

- The CICS log files are not cleared unless a region is shut down normally. Repeated abnormal shutdowns can lead to a growth in the size of the log file.
- Do not use a Network File System (NFS) for the CICS log.

## Using `cicsexport` and `cicsimport` for backup and recovery

You can use the `cicsexport` and `cicsimport` commands to back up and restore some of the data on the region directory. This action is useful only for cold starts because the runtime database is not backed up and restored with these commands.

The `cicsexport` command copies the following data:

- The cold start (permanent) database that contains the resource definitions for the region. If you require any of the region definitions to be shared between the region that you are copying and the region to which you wish to restore the copy, consider maintaining a single physical copy of the definitions and using *symbolic links* to provide an up-to-date copy in the second region.
- Conversion templates.
- All programs in the region's **bin** directory.
- All maps in the region's **bin** directory.
- The contents of the region's data directory.

For example, to archive the resource definitions for regionA to an output file named REG1ARCH, enter the command:

```
cicsexport -r regionA -o REG1ARCH
```

Then to restore the resource definitions for regionA, enter the command:

```
cicsimport -r regionA -i REG1ARCH
```

Before attempting to use the **cicsexport** and **cicsimport** commands, review the information about them in the *TXSeries for Multiplatforms Administration Reference*.

If the CICS resource definitions cannot be re-created from backup or archive, you can reconfigure them by using the procedures described in Chapter 4, “Working with a file manager in a CICS region,” on page 23.

## Enabling recovery of the CICS subsystem entries

Many of the configuration tasks described in Chapter 4, “Working with a file manager in a CICS region,” on page 23 create operating system subsystem entries.

The subsystem entries for CICS components are updated as a result of an administrative task and require backup only occasionally. See your operating system documentation for information about where the subsystem entries are located and how to back them up.

If the CICS subsystem entries cannot be recovered from backup, you can re-create them by reconfiguring the CICS components, as described in Chapter 4, “Working with a file manager in a CICS region,” on page 23.

## Enabling recovery of the CICS user IDs

The operating system user ID information for CICS components is updated as a result of an administrative task and requires only occasional backup. See your operating system documentation for information about where the user ID information is located and how to back it up.

If the CICS user IDs cannot be recovered from backup, you can re-create them by reconfiguring the CICS components, as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.

## Full recovery after a complete system failure

A full recovery is required when your CICS system is completely disabled; for example, when the hardware is replaced. This procedure assumes that you have the installation media for DB2 (if you are using DB2 for queue and file management), and CICS, and that you have the backup tapes that you made earlier.

Do the following steps.

1. Install CICS.

If you have to replace the hardware, or if you lose a disk that contains the CICS code, reinstall them in the same way that you originally installed them.

2. Recover queue and file management.

Use one of the following procedures, depending on the product you use:

- Recover SFS.

If you have not lost the operating system logical volumes for the SFS server, you can autostart the SFS server. After you autostart the SFS server, look at the message file `/var/cics_servers/SSD/cics/sfs/baseName/msg`.

A message file that is created for a normal autostart contains a message that is similar to the following:

```
1 A sfs: Initialized ... Thu Aug 19 07:55:54 1994
```

Use the contents of the message file to diagnose any outstanding problems. Possible problems and solutions follow:

- The logical volumes for the SFS server no longer exist. See “Recovering SFS queues and file management” on page 246, which describes how to recover the SFS logical volumes.

If you cannot resolve the errors in the message file, you can attempt a cold start of the SFS server. However, use a cold start only as a last resort because this discards all files that are located on the SFS logical volumes.

- Recover DB2.

If it is required that you reconfigure the database, see “Administration of file and queue management on DB2” on page 31.

See the *DB2: Administration Guide* for information about restoring the data.

### 3. Recover CICS service.

Restore the CICS permanent and runtime resource definition files that are in **/var/cics\_regions/regionName/database**.

If a cold start of the region is not required, restore the CICS log files that are in **/var/cics\_regions/regionName/log**.

If you were able to autostart the SFS server, attempt to warm start the region. If you cold started the SFS server, you must also cold start the region. However, if you cold start the region, the runtime resource definition files and the CICS log files are discarded.

The CICS region reports any problems to its region console file, which is located in **/var/cics\_regions/regionName/console.nnnnnnn**.

Review the message files for errors. The **console.nnnnnnn** files are generated in sequence. The **console.nnnnnnn** file is in **/var/cics\_regions/regionName/data**. Every time a new CICS region starts, the number of the **console.nnnnnnn** file is incremented by one. The size of this file is controlled through the **MaxConsoleSize** attribute of the Region Definitions. Setting **MaxConsoleSize** to 0 (zero) means that the **console.nnnnnnn** file will grow until it reaches the limits of your system. Setting it to a specific size means that the first file will be closed and a new file opened with the next available number. Two messages are issued that tell you when the file is closed and the name of the new file. The **cicstail** command automatically continues to tail the new file.

**Note:** If your CICS region is running and the SFS server stops for any reason, files are marked as DISABLED. If you take administrative action to restore the SFS server and the region is still running, you must set the files to enabled; for example, by using CEMT SET ENABLED. If many files are to be reset, it can be simpler to restart CICS automatically.

## Suggestions for backing up your systems

How you plan for recovery depends on your requirements, the configuration of your CICS systems, and the resources that are available to you.

Review the following scenarios to help you decide how to design your system for the level of recovery that you require.



## **A nonproduction CICS system**

If you are using a nonproduction CICS system, for example, for development and testing, dynamic recovery of your running applications is probably less important than it is in a production system. Therefore, mirroring can be unnecessary.

Offline backups of your nonproduction system are required if you need to restore a nonproduction system to the original configuration. However, to use offline backups effectively for warm starting a system to a previous state, the backups must be taken when both the region and the file manager are shut down. Otherwise, data such as transient data queues that are stored on the file manager can be unsynchronized with the CICS log data that is stored in the region file system. In that case, a cold start can be required to bring up the region.

A CICS cold start discards the data that is in the CICS log, any data that is in the runtime resource definition database, and any data that is in the CICS queues. An SFS cold start discards all files and their contents that are maintained by the SFS server. No concept exists of a DB2 cold start; restarting DB2 maintains all the data that is held in the database. However, it has no effect on queues and files that DB2 manages.

The data that is in the runtime resource definition database can be important, especially in a test environment where resource definitions are tested before they are committed to the permanent resource definition database.

Cold starting an SFS requires that you re-create the files on the SFS logical volumes. If you must cold start the SFS server, see “Administration of file and queue management on an SFS server” on page 23 for information about re-creating the files.

## **Production CICS systems**

The method that you use to prepare for recovery of a production CICS system depends on how dynamic the system is, the application data on the system, and how current the system needs to be at the time that it is restored.

A region that operates as a terminal concentrator, shipping all transactions to one or more remote regions for action, or a region that runs transactions requiring a large amount of computation but only a limited amount of data that is managed by a remote region, can possibly use backups without mirroring. However, unless you use mirroring, you cannot restore the system to its state at the time of the failure. The system can be restored only to its state at the time of the last backup. This limitation is adequate only if the in-flight transaction states of the last backup can be used.

For production systems with data, whether or not you use mirroring depends on how much data you are willing to lose. If you can afford to restore your data to the state that it was in at the time that the last offline backup was taken, mirroring is not required for backup of data.

## **Improving your ability to restore services**

By testing backup and recovery procedures, you can estimate how long it takes to restore service, both after a total failure in which the disks and processors are completely destroyed, and after various lesser failures.

If the length of time to restore service after some of the lesser failures is unacceptable, you can redesign your backup strategy to provide a better recovery.



In a transaction processing network that has more than one administrative domain (for example, a network joining two or more regions), each region can have its own file manager. In this case, transactions can update files in both file managers under a common syncpoint. In this case, it is important to be able to recover the CICS log. One reason for this is that if a transaction is between the prepare and commit phases of syncpoint processing when a failure occurs, the unaffected file manager continues to hold locks that are associated with the affected transaction until they are manually released or until the CICS log in the affected system is restored to the state that it was in at the time of the failure. To manually release locks in DB2, use the LIST INDOUBT TRANSACTION command as described in the *DB2: Command Reference*.

### **Using mirrored volumes**

To help you to restore CICS services quickly, you can use sequentially mirrored volumes for the CICS log, the file manager log, the data, and the configuration information. If you do not have these volumes on mirrored volumes and you suffer a disk failure, all files that are on the affected volume are unavailable until you stop the file manager to perform recovery. Also, all the files that are on the affected file manager are unavailable while the file manager is in administrative mode and you are performing recovery.

Typically, you must allocate a relatively large amount of disk space to file manager volumes, in comparison with other uses of disk space, so the cost of disks for mirroring can be significant.

### **Using a second network interface card and extra network cabling**

Using a second network interface card and extra network cabling avoids interruption of service because a defective network interface card or another defective device on the network prevents communications on the network.

### **Using a second processor as a reserve**

Using a second processor as a reserve avoids interruption in service because of a machine hardware fault. Depending on the design of your application, you can normally run a region and a file manager on both processors.

You can also use a second processor and two sets of twin-tailed external disks. With such a system, in which all configuration information and application data is on external disks and mirrored, and in which both processors are directly connected to both disks, you have the following features:

- Normal condition: A region and a file manager on each machine. Transactions on either region have access to data managed by both file managers, and the workload is shared between the machines.
- Disk or controller failure: No service interruption occurs because mirroring ensures that the disks continue to be available.
- Processor failure: Move the disks that were attached to the failed processor to the working processor. Mount the file system that contains the configuration information for the file manager and region that were on the failed processor. Autostart the file manager and the region that were running on the failed processor on the working processor.

A service interruption occurs to those transactions that are running in one of the regions and to those transactions that are using one of the file managers, but service is rapidly restored.

Appropriate file systems, DB2 instances, and SFS volumes are set up to ensure that the vital items are placed as required on the twin-tailed disks.

**On the AIX platform.** CICS supports the RS/6000® HA-S70 Advanced Cluster Server to provide a high availability solution for business-critical and e-business applications. For more information about implementing this solution, see the IBM RS/6000 Web site ([www.rs6000.ibm.com](http://www.rs6000.ibm.com)).

## Scheduling offline backups

Offline backups can be used for configuration information; that is, all information about your CICS system except the CICS log and the data that is stored on the file manager logical volumes. Offline backups can also be used for those conditions in which the restoration of applications to the point of failure is not required.

Schedule your backups in accordance with the amount of work that it is acceptable to lose. For example, if you can afford to lose the information for a day's work, schedule the offline backups for once a day. The need to shut down the region and the file manager affects your decisions about the timing and frequency of the backups.

When you use an SFS server for queue and file management, you must schedule periodic backups of the SFS volume, which can be large and awkward to copy. Consider setting the parameters for the file size and the number of files so that each execution of the backup tool backs up part of an SFS volume instead of the whole volume. In this way, the backup tool produces files of a manageable size, but you need to ensure that it takes backups quickly and frequently. This is known as *fuzzy* or *incremental* backup.

For example, you can set the backup parameters such that the backup size is one-eighth of the whole SFS volume. Any eight successive backup files constitute a complete backup of the volume.

During recovery, the complete backup of the SFS volume is loaded, and the information in the log volumes and archive files is used to redo any changes that have occurred since the start of the backup; that is, since the time that the first of the eight incremental backups started. The result is to bring the SFS volume up-to-date. If the backup of the SFS volume is old, much log data is available to scan and act on. If the backup of the SFS volume is newer, less log data is needed, so recovery occurs more quickly.

If you are using DB2 for queue and file management, refer to the BACKUP DATABASE utility information in the *DB2: Administration Guide*.

The incremental backup mechanism ensures that all file managers and regions are restored to a state that is consistent for all transactions and consistent with the state of the log. This means that when the backup is restored:

- The effects of all transactions that were committed before the backup are redone, if necessary.
- The effects of all transactions not yet begun, or begun and not yet committed, are undone.

These are the desirable effects for a transaction processing system, but they do not allow for recovery from the effects of a defective application program that possibly incorrectly adds, changes, or deletes records in a file and then commits the transaction.



---

## Part 4. Appendixes



---

## Appendix A. Using the AIX System Management Interface Tool (SMIT)

**On the AIX platform.** The System Management Interface Tool (SMIT) can also be used to configure CICS regions and resources. To use SMIT to create, modify, start or destroy regions or file servers, you must be:

- Logged onto the AIX operating system with Administrator privileges

CICS is invoked with the following commands:

- **smit cics:** This command checks the DISPLAY environment variable before invoking SMIT. If this environment variable is set, the Motif version of the SMIT interface is invoked. Otherwise, the character-based interface is invoked.
- **smitty cics:** This command invokes the character-based interface.

This document uses the character-based interface to illustrate its examples.

**Attention:** Do not directly edit the resource definition file.

The initial panel (character-based interface) is shown in Figure 12.

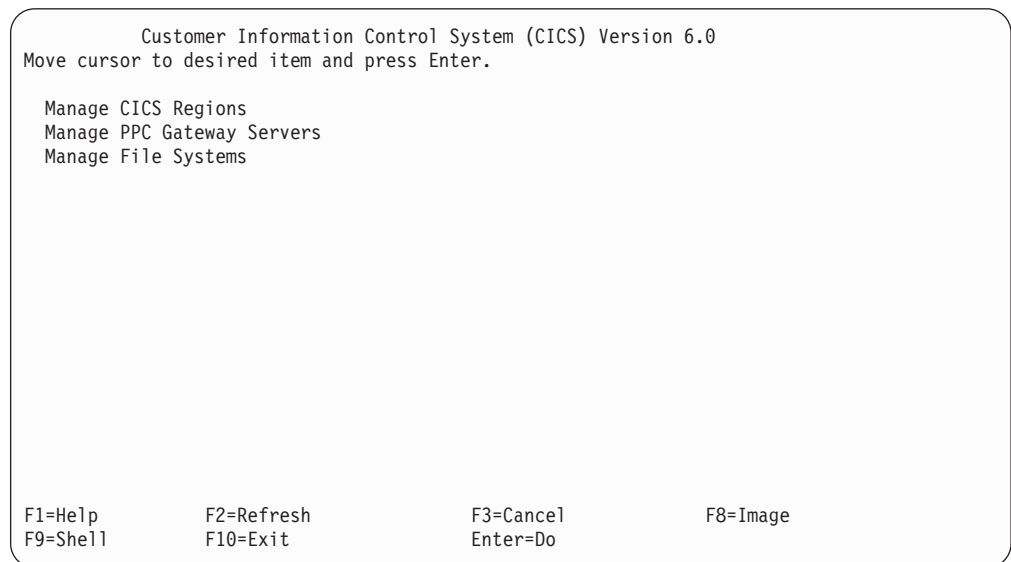


*Figure 12. The Customer Information Control System (CICS) Version 6.0 panel*

The initial panel provides the following options:

- **Manage CICS Regions:** To create, change, or delete resource definitions that are associated with a region and the region resources.
- **Manage PPC Gateway Servers:** To create, change, or delete resource definitions that are associated with the PPC Gateway server.
- **Manage File Systems:** To create, change, or delete resource definitions that are associated with the file manager.
- **F1 (Help):** To display the help information for this panel.
- **F2 (Refresh):** To refresh the screen.

- **F3 (Cancel):** To exit from SMIT. On subsequent panels, F3 returns you to the previous panel.
- **F8 (Image):** To save the screen image in a log file (placed onto your home directory).
- **F9 (Shell):** To create a shell.
- **F10 (Exit):** To exit from SMIT.
- **Enter:** To perform the operation that is displayed on the panel. In this case, the item is highlighted.

---

## Metacharacter restrictions

On an insert, update, read, and delete operation for the permanent database, CICS checks the bytes that form the database key, to ensure that the key does not contain any of the bytes that operating system software such as SMIT cannot handle. When it encounters any of the following bytes, CICS displays an error message and rejects the database operation:

- 0 through 32 decimal (NULL through space character)
- 35 decimal (Hash [#] )
- 39 decimal (Single quote ['] )
- 58 decimal (Colon [:])
- 61 decimal (Equal sign [=] )
- 92 decimal (Backslash [\])
- 127 through 255 decimal (DEL through all 8-bit characters)

In addition, for all insert, update, read, and delete permanent database operations, and on a runtime database insert operation, CICS performs class-specific validation of the bytes that form the database key for transactions, programs, terminals, communications, users, files, temporary storage, and transient data.

If you are using the CICS resource management commands (such as **cicsadd**, **cicsupdate**, or **cicsdelete**) and want to enter calls that include metacharacters, such as quote ("), backslash (\), dollar (\$), or asterisk (\*), you must use the shell escape mechanism. For example, to enter the value **aa**, you can use **a\\a** or **a"a**.

SMIT does not support the use of metacharacters in the SMIT fields. If you need to specify a value that includes a metacharacter, use the CICS resource management commands from the operating system command prompt. For information about the byte values that are allowed in the key, see the *TXSeries for Multiplatforms Administration Reference*.

The SMIT interface and the CICS resource management commands, such as **cicsupdate**, **cicsadd**, **cicsdelete**, and **cicsget**, run outside the CICS runtime environment and do not need to interface with a CICS region to perform this service. The **cicsinstall** command requires that CICS is running, because **cicsinstall** needs to access the CICS runtime database.

---

## Using SMIT to add resource definitions

To add a resource definition to the region by using SMIT:

1. Enter **smitty cics**. The **Customer Control Information System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.



If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.

3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the resource class that you want to add.

**Note:** For Region and Monitoring definitions, you can update only an existing resource definition.

6. Select **Add New**.

Use the Select a Model dialog box to base the new definitions on an existing model or to create a new definition from default values. You can modify these default values by using the **cicsupdate** command.

7. SMIT displays the definitions for the new resource. Specify the attribute values and press **Enter**. SMIT now updates the appropriate definitions automatically.

For information about how to use SMIT fast path to perform this task, see “Using the SMIT fast path to add a resource” on page 264.

## Using SMIT to add a user

When adding a user with SMIT, complete the following fields. (You cannot change these fields by using the **cicsupdate** or **cicsadd** commands.)

1. Enter **smitty cics**. The **Customer Control Information System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.  
If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.
3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the **Users** resource class from the list of resource classes.
6. Select **Add New**.
7. Specify values for all the desired fields. The following fields are of specific interest:
  - **User Identifier**. Enter the user ID for the new user.
  - **Ignore errors?** Set this field to **yes** to allow an operation that was partially successful to be resubmitted. Normally, if an operation to add an item finds that the item already exists, the operation handles this as an error. Use the **Tab** key to toggle between the values.
8. When all the desired components have been entered, press **Enter**. SMIT adds the appropriate definitions automatically.

---

## Using SMIT to change a resource definition

To modify a resource definition in a CICS region by using SMIT:

1. Enter **smitty cics**. The **Customer Control Information System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.

If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.

3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the resource class that you want to modify.
6. Select **Show/Change** and then select a particular resource for display or update.
7. Change all the required components and press **Enter**. SMIT updates the appropriate definitions automatically.

For information about how to use SMIT fast path to perform this task, see “Using SMIT fast path to change a resource definition” on page 265.

## Using SMIT to modify a user

When you modify a user with SMIT, complete the following fields. (You cannot change these fields by using the **cicsupdate** or **cicsadd** commands.)

1. Enter **smitty cics**. The **Customer Control Information System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.  
If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.
3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the **Users** resource class from the list of resource classes.
6. Select **Show/Change**.
7. Change all the desired components. The following components are of specific interest:
  - **New User Identifier**. Enter the new user ID for the user.
  - **User Identifier**. Enter the user ID for the user whose definition is to be modified.
8. When all the desired components have been entered, press **Enter**. SMIT updates the appropriate definitions automatically.

---

## Using SMIT to view a resource definition

To view a resource definition in the CICS region by using SMIT:

1. Enter **smitty cics**. The **Customer Information Control System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.  
If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.
3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the resource class that you want to display or change.
6. Select **Show/Change**.

7. SMIT displays the values of all the attributes for the chosen resource definition. For further information, read “Changing a resource definition in the permanent, autostart, and runtime databases” on page 117.

For information about how to use SMIT fast path to perform this task, see “Using SMIT fast path to inquire about a resource” on page 265.

---

## Using SMIT to delete a resource definition

Perform these steps to delete a resource from the CICS region by using SMIT:

1. Enter **smitty cics**. The **Customer Information Control System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.  
If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.
3. Select **Define Resources for a CICS Region**.
4. Select **Manage Resources**.
5. Select the resource class that you want to delete.
6. Select **Delete** and use **F7** to select one or more definitions from the list. Press **Enter** when you have completed the selection.
7. SMIT displays the resource definitions that are to be deleted. You can now choose:
  - **Delete**: Delete the definition from the permanent database.
  - **Uninstall**: Delete the definition from the runtime database.
  - **Both**: Delete the definition both from the permanent database and from the runtime database.
8. Press **Enter**. SMIT updates the appropriate database.

For information about how to use SMIT fast path to perform this task, see “Using SMIT fast path to delete a resource definition” on page 265.

---

## Using SMIT to refresh the region with new resource definitions

If you change or add a resource definition, for example, as described in “Using SMIT to add resource definitions” on page 260 and “Using SMIT to change a resource definition” on page 261, and you want the new definitions applied to the running region, you can use SMIT to refresh the runtime database. You can refresh the following:

- Resource definitions from a group or groups
- Resource definitions from the permanent database that have the **ActivateOnStartup** attribute set to **yes**

To use SMIT to refresh the runtime resource database:

1. Enter **smitty cics**. The **Customer Control Information System (CICS) Version 6.0** screen is displayed.
2. Select **Manage CICS Regions**.  
If necessary, select **Change Working CICS Region**, and select the appropriate region from the list. When the command status shows **ok**, use **F3** to return to the **Manage CICS Regions** screen.
3. Select **Define Resources for a CICS Region**.

4. Select **Install Group(s)**. SMIT displays a list of available regions. When you select one, SMIT displays a panel entitled Select Groups to Install, listing all the available groups in the region.
5. Use **F7** to select the group or groups that you want to install, and press **Enter**. SMIT displays a panel confirming the choice you have made, and when you press **Enter**, the installation occurs.

If you attempt to install permanent resource definitions from SMIT, SMIT always produces messages indicating which resources, if any, **cicsinstall** has installed, including the total number of resources installed.

For information about how to use SMIT fast path for this task, see “Using SMIT fast path to install a group of resources” on page 265.

---

## Using SMIT fast path to administer resources

The SMIT fast path facility allows you to go directly to a particular panel or menu from the operating system command line instead of through the panel hierarchy. The SMIT fast path enables resource definitions to be viewed, added, modified, or deleted.

To use the fast paths, you must set and export the environment variable **CICSREGION** to the name of the region.

The following list identifies the classes of resource definition on which these operations can be performed:

- **cd**: Communication Definitions
- **fd**: File Definitions
- **jd**: Journal Definitions
- **pd**: Program Definitions
- **td**: Transaction Definitions
- **tdd**: Transient data queue Definitions
- **tsd**: Temporary storage queue Definitions
- **ud**: User Definitions
- **wd**: Terminal Definitions
- **xad**: Product Definitions
- **gd**: Gateway Definitions
- **ld**: Listener Definitions

See the following sections for more information:

- “Using the SMIT fast path to add a resource”
- “Using SMIT fast path to change a resource definition” on page 265
- “Using SMIT fast path to install a group of resources” on page 265
- “Using SMIT fast path to inquire about a resource” on page 265
- “Using SMIT fast path to delete a resource definition” on page 265

### Using the SMIT fast path to add a resource

To use the SMIT fast path to add a resource definition, enter the command **smitty cicsaddxx** on the operating system command line, where **xx** is the two- or three-character mnemonic for the resource class. See “Using SMIT fast path to administer resources” for the list of resource classes.

For example, type **smitty cicsaddfd** to add File Definitions (FD).

The appropriate SMIT panel is displayed. Go to step 6 on page 261 in “Using SMIT to add resource definitions” on page 260.

## Using SMIT fast path to change a resource definition

To use the SMIT fast path to modify a resource definition, enter the command **smitty cicsupdatexx** on the operating system command line, where *xx* is the two- or three-character mnemonic for the resource class. See “Using SMIT fast path to administer resources” on page 264 for the list of resource classes.

For example, type **smitty cicsupdatetd** to update a transaction definition.

The appropriate SMIT window is displayed. Go to step 6 on page 261 of “Using SMIT to change a resource definition” on page 261.

## Using SMIT fast path to inquire about a resource

To use the SMIT fast path to inquire about a resource, enter the command **smitty cicsupdatexx** on the operating system command line, where *xx* is the two- or three-character mnemonic for the resource class. See “Using SMIT fast path to administer resources” on page 264 for the list of resource classes.

For example, type **smitty cicsupdatepd** to obtain information from a program definition.

The appropriate SMIT window is displayed, and you continue as described in Step 6 on page 262 of “Using SMIT to view a resource definition” on page 262.

## Using SMIT fast path to delete a resource definition

To use the SMIT fast path to delete a resource definition, enter the command **smitty cicsdeletexx** on the operating system command line, where *xx* is the two- or three-character mnemonic for the resource class. See “Using the SMIT fast path to add a resource” on page 264 for a list of resource classes and their associated mnemonics.

For example, type **smitty cicsdeleteud** to add a user definition.

The appropriate SMIT window is displayed. Go to step 6 on page 263 of “Using SMIT to delete a resource definition” on page 263.

## Using SMIT fast path to install a group of resources

To use the SMIT fast path method of installing a group of resource definitions, enter the command **smitty cicsinstallgr** at the operating system command prompt.

---

## Using SMIT to add a region

To use SMIT to add a region:

1. Enter **smitty cics**.
2. When the Customer Information Control System (CICS) Version 6.0 panel is displayed, select **Manage CICS Regions**.
3. On the Manage CICS Regions panel, select **Create (Import) a CICS Region**.  
When the next panel is displayed:

- a. Type the name of the new region in the **Name of Region to be Created** field.
- b. If you are creating a region from the default resource definitions, press **Enter**.
- c. If you are creating a region from an existing set of region resource definitions, type the path of the region in the **Input file** field and press **Enter**.

The region definitions are placed in the directory `/var/cics_regions/regionName` where *regionName* is the region name you selected.

4. Return to the Manage CICS Regions panel and select the **Change Working CICS Region** option. The Select a Region window is displayed with the new region listed.
5. Select the new region. The Command Status panel is displayed. When **Command: OK** is shown in the upper left corner, use **F3** to return to the Manage CICS Regions panel.
6. Select **Configure CICS Resources on a File Server**. When the next panel displayed, choose the type of file server to be configured and configure the file server resources. Use **F3** to return to the Manage CICS Regions panel.
7. Select **Define Resources for a CICS Region**.
8. In the Define Resources for a CICS Region panel, select **Manage Resource(s)**.
9. When the next panel is displayed, select the resource class that you want to work with.

The classes of resource definitions that you can work with are listed in “Identifying resource definition classes” on page 5.

You can set up symbolic links between the region definition of one region and resource definitions in the permanent database of another region. See “Configuring for shared resource definitions” on page 125 for information on how to specify shared resources.

---

## Using SMIT to delete a region

To use SMIT to delete a region:

1. Enter **smitty cics**
2. When the Customer Information Control System (CICS) Version 6.0 panel is displayed, select **Manage CICS Regions**.
3. Select **Destroy a CICS Region**.
4. In the Destroy a CICS Region panel, type the name of the region that you want to delete in the **Name of Region** field, and press **Enter**.
5. In the Are You Sure? panel, press **Enter** to continue.

All resource definitions, except those defined for the SFS server, are deleted.

---

## Using SMIT to add an SFS server

To use SMIT to add an SFS server:

1. Enter **smitty cics**.
2. When the Customer Information Control System (CICS) Version 6.0 panel is displayed, select **Manage Filesystem**.
3. On the **Manage Filesystem** panel, select **Manage CICS SFS Servers**.
4. On the **Manage CICS SFS Servers** panel, select **Define CICS SFS Server**.

5. On the **Define CICS SFS Servers** panel, select **Create**. When the next panel is displayed, type the model SFS Server Identifier and press **Enter**. If you want to use the default values, press **Enter**.
6. On the **Create CICS SFS Servers** panel:
  - a. Type the name of the new SFS Server Identifier in the **SFS Server Identifier** field.
  - b. Type the short name used for the SFS Server in the **Short name used for SRC** field.
  - c. Type the created AIX user ID in the **AIX user ID for server** field. If you are using the default short name as the AIX user ID press **Enter**. This AIX user ID should exit on the system.

The SFS Server definitions are placed in the file  
/var/cics\_servers/SSD/SSD.stanza.

The physical storage devices must be prepared before starting the SFS server (see “Preparing the physical storage devices for the SFS server” on page 274).

For more information see “Defining an SFS server by using the cicssfscree command” on page 275.

---

## Using SMIT to add a PPC Gateway server

To use SMIT to add a PPC Gateway server:

1. Enter **smitty cics**.
2. When the Customer Information Control System (CICS) Version 6.0 panel is displayed, select **Manage CICS PPC Gateway Server**.
3. On the **Manage CICS PPC Gateway Servers** panel, select **Define CICS PPC Gateway Servers**.
4. On the **Define CICS PPC Gateway Servers** panel, select **Create**. When the next panel is displayed, type the PPC Gateway Server Identifier and press **Enter**. If you want to use the default values, press **Enter**.
5. On the **Create CICS PPC Gateway Servers**:
  - a. Type the name of the new PPC Gateway Server Identifier in the **PPC Gateway Server Identifier** field.
  - b. Type the short name used for PPC Gateway Server in the **Short name used for SRC** field.
  - c. Type the created AIX user ID in the **AIX user ID for server** field. If you are using the default short name as the AIX user ID press **Enter**. This AIX user ID should exit on the system.

The PPC Gateway Server definitions are placed in the file  
/var/cics\_servers/GSD/GSD.stanza. For more information see the information about Using CICS commands to manage a PPC Gateway server in the *TXSeries for Multiplatforms Intercommunication Guide*.





---

## Appendix B. Using CICS resource management commands for system management

The CICS control program (**cicscp**) command programs are the recommended method for setting up CICS. However, when you are working with a troublesome configuration, you can more rapidly identify problem areas by using the CICS resource management commands, which underlie the **cicscp** command programs. The CICS resource management commands enable you to set up each CICS component step-by-step. The section, “The CICS resource management commands used by the **cicscp** command” provides an overview of the CICS resource management commands that are called from within the **cicscp** command programs.

“Commands used to set up a CICS system” on page 270 provides a list of CICS resource management commands that are used during the creation of a CICS configuration. “Commands used to prepare a region for production” on page 270 provides list of CICS resource management commands that are used to prepare a CICS system for production use. Table 39 on page 271 shows the correct sequence for combining these CICS resource management commands to achieve specific goals.

**On the AIX platform.** The System Management Interface Tool (SMIT) can also be used to perform some of these tasks.

---

### The CICS resource management commands used by the **cicscp** command

The **cicscp** command invokes the following underlying commands to create and start the region and the file manager:

- To create a new region using the default region definitions, the **cicscp create region** command calls the **cicsdefault** command.
- To import region definitions from an existing customized region, the **cicscp create region** command is invoked with the **-i** option (for import), and the file name of the file that is holding the exported region is supplied. In that condition, it is the **cicsimport** command that is invoked by the **cicscp create region** command.
- To create, configure, and start the SFS server, the **cicscp** commands invoke the following CICS resource management commands:
  - **cicssfscree**
  - **cicssf**
  - **cicssfscnf**
- To configure a DB2 database for file management, the **cicscp** command invokes the **cicsdb2cnf** command. (A DB2 database must be created and started before the **cicsdb2cnf** or **cicscp** command is used.)
- To start the region, the **cicscp** command invokes the following command, depending on the operating system:
  - **On the AIX platform.** The **cicscp** command invokes the **startsrc** command.
  - **On the HP-UX or Solaris platforms.** The **cicscp** command invokes the **cicsstart** command.

## Commands used to set up a CICS system

The following CICS resource management commands are used to set up a CICS region:

- **cicsdefault**: Create a copy of the default region. Use this command to create a new region from an archive copy of the default CICS region.
- **cicsexport**: Export CICS resource definitions. Use this command to make an archive copy of all the resource definitions that are in an existing CICS region. You use the **cicsexport** command in conjunction with the **cicsimport** command.
- **cicsimport**: Import CICS resource definitions. Use this command to restore an archive copy of the resource definitions from an existing CICS region. You use the **cicsimport** command in conjunction with the **cicsexport** command.
- **cicsdefaultservers**: Create default resource files for servers. Use this command to create the file system and default resource definition files for the SFS and PPC Gateway servers.
- **cicssfscscreate**: Create a CICS SFS server. Use this command to create the resource definitions for an SFS server that is used by CICS.
- **cicssfscs**: Start a CICS SFS server. Use this command to start an SFS server with the parameters that are required for CICS.
- **cicssfscsconf**: Configure a CICS SFS server. Use this command to configure an SFS server for a region by adding the CICS queues to it.
- **cicsdb2conf**: Configure DB2. Use this command to configure a DB2 database for a region by adding the CICS queues to it.
- **cicsstart**: Start a CICS region. Use this command to start the region.

To use these commands in the correct sequence, follow the instructions that are given in Table 39 on page 271.

## Commands used to prepare a region for production

The CICS resource management commands are used to prepare a region for production use. After your region is set up, you can use the following commands to define the resources to the region:

- **cicsadd**: Add CICS resource definitions. Use this command to add a definition for a resource.
- **cicsupdate**: Modify CICS resource definitions. Use this command to change a definition for a resource.
- **cicsdelete**: Delete CICS resource definitions. Use this command to delete a definition for a resource.
- **cicsget**: Get CICS resource definition information. Use this command to view a definition for a resource.
- **cicsinstall**: Install CICS resource definitions. Use this command to install the groups of resource definitions from the permanent database into the autostart and runtime databases.
- **cicssdt**: SFS diagnostic tool. Use this command to add the data files to the SFS server.
- **cicsddt**: DB2 diagnostic tool. Use this command to add the data files to DB2.

To use these commands in the correct sequence, follow the instructions that are given in Table 39 on page 271.

## Combining the CICS resource management commands to set up CICS

Table 39 shows how to combine the CICS resource management commands to do some common system administration tasks.

*Table 39. CICS system management tasks and the CICS commands to complete them*

| If you want to...                                                                 | Refer to...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Set up a region that uses a local SFS server for CICS queue and file management.  | <p>In this sequence:</p> <ol style="list-style-type: none"> <li>1. "Creating the region's default user ID" on page 272.</li> <li>2. "Using the cicsdefault command to define a region" on page 272 or "Defining a region using the cicsexport and cicsimport commands" on page 273.</li> <li>3. "Creating the user ID for the SFS server" on page 273.</li> <li>4. "Preparing the physical storage devices for the SFS server" on page 274.</li> <li>5. "Defining an SFS server by using the cicssfscscreate command" on page 275.</li> <li>6. "Starting an SFS server by using the cicssfscs command" on page 276.</li> <li>7. "Setting up the queues by using the cicssfscsconf command" on page 276.</li> <li>8. "Starting a region from the command line" on page 280.</li> </ol> |
| Set up a region that uses a remote SFS server for CICS queue and file management. | <p>In this sequence:</p> <ol style="list-style-type: none"> <li>1. "Creating the region's default user ID" on page 272.</li> <li>2. "Using the cicsdefault command to define a region" on page 272 or "Defining a region using the cicsexport and cicsimport commands" on page 273.</li> <li>3. "Setting up the queues by using sfsadmin commands" on page 277.</li> <li>4. "Starting a region from the command line" on page 280.</li> </ol> <p>These procedures assume that the SFS server was initially set up for a local region.</p>                                                                                                                                                                                                                                             |
| Set up a region that uses a DB2 database for CICS queue and file management.      | <p>In this sequence:</p> <ol style="list-style-type: none"> <li>1. "Creating the region's default user ID" on page 272.</li> <li>2. "Using the cicsdefault command to define a region" on page 272 or "Defining a region using the cicsexport and cicsimport commands" on page 273.</li> <li>3. "Preparing for DB2 file and queue management" on page 37.</li> <li>4. "Configuring resources and DB2 with cicsdb2conf" on page 63.</li> <li>5. "Starting a region from the command line" on page 280.</li> </ol>                                                                                                                                                                                                                                                                      |

Table 39. CICS system management tasks and the CICS commands to complete them (continued)

| If you want to...                                     | Refer to...                                                                 |
|-------------------------------------------------------|-----------------------------------------------------------------------------|
| Shut down the region from the command line.           | "Shutting down the region from the command line" on page 281.               |
| Shut down an SFS server by using <b>cicssfsshut</b> . | "Shutting down an SFS server by using the cicssfsshut command" on page 282. |
| Delete a region by using <b>cicsdestroy</b> .         | "Using the cicsdestroy command to delete a region" on page 282.             |
| Delete an SFS server by using <b>cicssfsdestroy</b> . | "Using the cicssfsdestroy command to delete an SFS server" on page 283.     |

## Creating the region's default user ID

Use this procedure to create the region's default user ID that is used for tasks that are not associated with a signed-on user ID and for the following conditions:

- When an attempt to sign on by using CESN fails
- When CRTE is used to route to another system and CESF is used to log off, the default user ID is used until CESN is used to log back on

A user is signed on to a CICS region as the default user ID when the following resource definitions are set:

- **AuthenticationService** is set to **CICS** in the Region Definitions (RD).

For intersystem requests, the following attributes are used in conjunction with the default user ID:

- The Communications Definitions (CD) **LinkUserId** attribute, and the TSL and RSL keys that are defined in the UD entry for the user ID that is specified with this attribute:
- The CD **RemoteSysSecurity** attribute
- The CD **TSLKeyMask** and **RSLKeyMask** attributes

To create the region's default user ID:

1. Define a user. Use the instructions that are given in "Setting up a CICS system for an end user" on page 14.
2. Set the RD **DefaultUserId** attribute to the key of the UD entry for that user.

**Note:** Because the default user ID is typically used as a sign-on identity for an unauthenticated user, take care when defining the UD entry to ensure that the **RSLKeyList** and **TSLKeyList** attributes allow access only to public transactions.

## Using the cicsdefault command to define a region

This procedure defines a region by using the **cicsdefault** command with the default region definitions. Review the *TXSeries for Multiplatforms Administration Reference* before using this procedure.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.
2. Use the following command example:

```
cicsdefault -r regionName -S
```

This command:

- Creates the region directory structure that includes the resource definitions from the default region. The ownership of the files in the region's directory structure is group **cics**.
- Sets the Region Definitions (RD) **NameService** attribute to **NONE** and the **AuthenticationService** attribute to **CICS**.

---

## Defining a region using the **cicsexport** and **cicsimport** commands

Use this procedure to create a new region from an existing set of region definitions. This procedure uses the **cicsexport** and **cicsimport** commands that are described in detail in the *TXSeries for Multiplatforms Administration Reference*. Review the information in those sections before using these commands.

**Note:** This procedure creates a copy of the region in **/var/cics\_regions**. Before you use this command, verify that you have enough space in **/var** for the copy.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.

2. Take a copy of an existing region, by using the **cicsexport** command. For example, the following command exports the regionA configuration to an archive file named **TEMPFILE**:

```
cicsexport -r 'regionA' -o 'TEMPFILE'
```

3. Use the **cicsimport** command to define the new region from the archive file. See the following example:

```
cicsimport -r 'regionB' -i 'TEMPFILE' -S
```

This command:

- Builds the following region directory structure:  
**/var/cics\_regions/regionB**  
The ownership of the files that are in the region's directory structure is group **cics**.
- Modifies the resource definitions from **TEMPFILE** with the name of the new region and sets the Region Definitions (RD) **NameService** attribute to **NONE** and the **AuthenticationService** attribute to **CICS**.

---

## Creating the user ID for the SFS server

This procedure describes how to create the operating system user ID for the SFS server. The requirements of the SFS server user ID are:

- If **cicscp** is used to configure the SFS server, the name of the SFS server user ID must be the first seven characters of the host name, preceded by the letter S. For example, if the host name is **hostABCD**, the name of the SFS server user ID must be **ShostABC**.

If CICS resource management commands (or SMIT on AIX) are used instead of **cicscp** to configure the SFS server, the name of the SFS server user ID must be **SFS\_SERV**.

- The primary group for the user ID must be **cics**.
- The home directory for the SFS server user ID must be **/var/cics\_servers/SSD/cics/sfs/serverId**

where *serverId* is the unique identifier for the SFS server. The default value that is used for *serverId* is the host name.

- Core files can be created on behalf of the SFS server user ID. To ensure that a full core file is captured, set the maximum size of the core file for the SFS server user ID to 81950 blocks.

Note that if the maximum size of the core file for the SFS server user ID is set to 81950 blocks and a core file is created, up to 40 MB of disk space can be required. For information about physical storage for the SFS server, see “Preparing the physical storage devices for the SFS server.”

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the operating system command for creating a user to create the SFS server user ID. (You can also use the operating system’s administration tool to do this task.) For example, if **cicscp** is used to configure the SFS server, and if the host name is **hostABCD**, you can use the following command:

**On the AIX platform.**

```
mkuser pgrp='cics' home='/var/cics_servers/SSD/cics/sfs/hostABCD \
 core='2097152' ShostABC
```

In this example, **cics** is the primary group, **/var/cics\_servers/SSD/cics/sfs/hostABCD** is the home directory, and **ShostABC** is the user ID.

**On the HP-UX or Solaris platforms.**

```
useradd -d /var/cics_servers/SSD/cics/sfs/hostABCD \
 -g cics -s /bin/ksh ShostABC
```

The Korn shell (**/bin/ksh**) is required for the **ShostABC** user.

3. If you want to be able to collect and read core dumps for the SFS server, ensure that the maximum core file size for the user is enough.
4. The user ID definitions must not come from the Network Information Service (NIS).

---

## Preparing the physical storage devices for the SFS server

The SFS server requires two logical volumes, one for the log data and one for the queue and file data. The size of the SFS server logical volumes must be at least 64 MB each. An additional 64 MB is required for each additional region.

The Structured File Server Definitions (SSD) **DataVolume** and **LogVolume** attributes are used to specify the names of the SFS logical volumes. The default values are **sfs\_shortName** and **log\_shortName**. The default value that is used for the SSD **ShortName** attribute depends on how you configure the SFS server:

- If you use the **cicscp** command to configure the SFS server, **ShortName** is set by default to the first seven characters of the host name, preceded by the letter S. For example, if the host name is **hostABCD**, then the default short name for the SFS server is **ShostABC**. The default names of the logical volumes are then **log\_ShostABC** and **sfs\_ShostABC**.
- If you use the **cicssfscreate** command to configure the SFS server, the **ShortName** attribute is set by default to **SFS\_SERV**. The default names of the logical volumes are then **log\_SFS\_SERV** and **sfs\_SFS\_SERV**. (This also the case if you have used SMIT on the AIX platform to do this task.)

If you use the default values when configuring the SFS server, you must use the default values when setting up the logical volumes.

To use this procedure:



- You must be logged in as **root** and **root** must be set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
- The SFS server user ID must be created as described in “Creating the user ID for the SFS server” on page 273.

Follow these steps to set up the SFS logical volumes:

#### On the AIX platform

1. Use operating system commands to create the logical volumes. (You can also use the system’s administration tool to do this task.) For example, if the host name is **hostABCD**, you can use the following commands:
 

```
mklv -y log_ShostABC rootvg 16
mklv -y sfs_ShostABC rootvg 16
```

 where **rootvg** is the volume group, and 16 is the recommended number of logical partitions. A logical partition is 4 MB.  
 This command places the logical volumes into the **/dev** directory.
2. Use the **chown**, **chmod**, and **chgrp** commands to set the access permissions and ownerships for the logical volumes.

#### On the HP-UX platform

1. Use the HP System Administration Manager (SAM) to create a file system on a separate logical volume. See your HP documentation for information on using SAM.
2. Use the **chown**, **chmod**, and **chgrp** commands to set the access permissions and ownerships for the logical volumes.

#### On the Solaris platform

1. See the details that are given in the *TXSeries for Multiplatforms Installation Guide* or refer to your operating system documentation for information about how to perform this procedure.
2. Use the **chown**, **chmod**, and **chgrp** commands to set the access permissions and ownerships for the logical volumes.

---

## Defining an SFS server by using the **cicssfscree** command

Use the **cicssfscree** command to do the following:

- Create the Structured File Server Definitions (SSD) for the SFS server, using either default values or values that are specified with the **cicssfscree** command.
- Create the subsystem entry for the SFS server.

The physical storage devices must be prepared before this procedure can be done. See “Preparing the physical storage devices for the SFS server” on page 274.

To define the SFS server:

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the **cicsdefaultservers** command to create the SFS server directory structure.
3. Issue the command:

```
cicssfscreate -S /./cics/sfs/serverId ShortName=shortName
```

In this example, the user ID and logical volume names are not specified. The default names for the logical volumes are **log\_shortName** and **sfs\_shortName**, and the default name for the SFS server user ID is the short name. If the physical devices were not set up to use these names (see “Creating the user ID for the SFS server” on page 273), use the **LogVolume** and **DataVolume** attributes to change the defaults. If the user ID was not set up with the short name (see “Preparing the physical storage devices for the SFS server” on page 274), use the **UserID** attribute to change the default. For example:

```
cicssfscreate /./cics/sfs/hostABCD \
 ShortName=ShostABC \
 ProtectionLevel=connect \
 LogVolume="accountlog" \
 DataVolume="accountdata" \
 UserID="accountsfs"
```

See the *TXSeries for Multiplatforms Administration Reference* for a complete description of all attributes that are used to define the SFS server to CICS.

---

## Starting an SFS server by using the **cicssfs** command

This procedure starts the SFS server so that the CICS queues and files can be added to the SFS logical volumes.

Before starting the server, ensure that the following procedures have been completed:

- “Creating the user ID for the SFS server” on page 273
- “Preparing the physical storage devices for the SFS server” on page 274
- “Defining an SFS server by using the **cicssfscreate** command” on page 275

**Attention:** Cold starting the SFS server destroys files on the SFS logical volumes. The following procedures tell you when to cold start and when to autostart.

To start the SFS server:

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the **cicssfs** command to start the SFS server. For example:

```
cicssfs /./cics/sfs/hostABCD
```

See the *TXSeries for Multiplatforms Administration Reference* for more information about the parameters that you use when you are starting an SFS server.

See also Chapter 16, “Starting up a CICS system,” on page 197.

---

## Setting up the queues by using the **cicssfsconf** command

This procedure uses the **cicssfsconf** command to add the files to the SFS logical volumes for CICS queue management.

Before adding the files, ensure that the following procedures have been completed:

- “Creating the user ID for the SFS server” on page 273
- “Preparing the physical storage devices for the SFS server” on page 274
- “Defining an SFS server by using the **cicssfscreate** command” on page 275
- “Starting an SFS server by using the **cicssfs** command”

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Ensure that the Region Definitions (RD) **DefaultFileServer** attribute is set to the name of the SFS server that is used for this region’s queue and file data. For example, if you have been preparing an SFS server named **./cics/sfs/hostABCD** for this configuration, ensure that the **DefaultFileServer** attribute is set to **./cics/sfs/hostABCD**.
3. Use the **cicssfscnf** command to add the queues to the region. For example:  

```
cicssfscnf wc regionA
```

where **wc** specifies that the SFS server is to be configured without changing any Region Definitions values. See the *TXSeries for Multiplatforms Administration Reference* for details about using this command.

**Note:** For information about how to protect the queues from access by unauthorized users, see “Authorizing user access to CICS resources” on page 99.

---

## Setting up the queues by using sfsadmin commands

Each region requires seven files to be added to the SFS server. Five files are required for managing the CICS temporary storage queue and transient data queue. Two files are required for the control of sending EXEC CICS START requests to other systems.

Before setting up the queues, ensure that the SFS server is set up and started for CICS use and that the region is configured.

Use this procedure to add the files:

1. From the host that has the region on it, log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the following command to gather information about the files.  

```
cicsget -c rd -r regionName
```

The **-c** option allows the resource definition class to be identified. In this case, the class is the Regions Definitions class, as indicated by the mnemonic **rd**.

The following list describes, by information category, the attributes that contain the relevant information and, where applicable, the **sfsadmin** command option to use when the file is created:

- The name of the SFS server for this region must be set with the **DefaultFileServer** attribute.
- Unexpired starts (nonprotected):  
**LocalQFile** (the name of the file)  
**LocalQIndex** (the primary index name)  
**LocalQMaxRecs** (the **-recordlimit** option)  
**LocalQPrePages** (the **-preallocate** option)  
**LocalQProtection** (the protection level)  
**LocalQVol** (the logical volume name)
- Unexpired starts (protected):  
**LocalQProtectFile** (the name of the file)  
**LocalQProtectIndex** (the primary index name)  
**LocalQProtectMaxRecs** (the **-recordlimit** option)

- LocalQProtectPrePages** (the **-preallocate** option)
  - LocalQProtectProtection** (the protection level)
  - LocalQProtectVol** (the logical volume name)
  - Logically recoverable transient data queue:
    - LocalTDQFile** (the name of the file)
    - LocalTDQIndex** (the primary index name)
    - LocalTDQMaxRecs** (the **-recordlimit** option)
    - LocalTDQPrePages** (the **-preallocate** option)
    - LocalTDQProtection** (the protection level)
    - LocalTDQVol** (the logical volume name)
  - Physically recoverable transient data queue:
    - PhysicalTDQFile** (the name of the file)
    - PhysicalTDQIndex** (the primary index name)
    - PhysicalTDQMaxRecs** (the **-recordlimit** option)
    - PhysicalTDQPrePages** (the **-preallocate** option)
    - PhysicalTDQProtection** (the protection level)
    - PhysicalTDQVol** (the logical volume name)
  - Nonrecoverable transient data queue:
    - NonRecTDQFile** (the name of the file)
    - NonRecTDQIndex** (the primary index name)
    - NonRecTDQMaxRecs** (the **-recordlimit** option)
    - NonRecTDQPrePages** (the **-preallocate** option)
    - NonRecTDQProtection** (the protection level)
    - NonRecTDQVol** (the logical volume name)
  - Nonrecoverable temporary storage queue:
    - NonRecTSQFile** (the name of the file)
    - NonRecTSQIndex** (the primary index name)
    - NonRecTSQMaxRecs** (the **-recordlimit** option)
    - NonRecTSQPrePages** (the **-preallocate** option)
    - NonRecTSQProtection** (the protection level)
    - NonRecTSQVol** (the logical volume name)
  - Recoverable temporary storage queue:
    - RecTSQFile** (the name of the file)
    - RecTSQIndex** (the primary index name)
    - RecTSQMaxRecs** (the **-recordlimit** option)
    - NonRecTSQPrePages** (the **-preallocate** option)
    - RecTSQProtection** (the protection level)
    - RecTSQVol** (the logical volume name)
3. Use the following steps to add the files to the SFS server:
    - a. From the host that has the SFS server on it, log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
    - b. Set the following environment variables:
      - CICS\_TK\_SERVER to `./cics/sfs/serverId`
      - CICS\_TK\_SFS\_SERVER to `./cics/sfs/serverId`
    - c. Use the **sfsadmin** command to create the files as shown:
      - Unexpired starts (nonprotected):
 

```
sfsadmin create clusteredfile regionNamecicsnlqfile \
 2 key byteArray 8 data varLenByteArray 32768 \
 cicsnlqidix -unique 1 key sfs_shortName \
 -preallocate 10 -recordlimit 1000000
```
      - Unexpired starts (protected):

```
sfsadmin create clusteredfile regionNamecicsplqfile \
 2 key byteArray 8 data varLenByteArray 32768 \
 cicsplqidx -unique 1 key sfs_ shortName \
 -preallocate 10 -recordlimit 1000000
```

- Logically recoverable transient data queue:

```
sfsadmin create clusteredfile regionNamecicstdqlgfile \
 2 itqkey byteArray 12 itqdata varLenByteArray 32768 \
 cicstdqlgidx -unique 1 itqkey sfs_ shortName \
 -preallocate 10 -recordlimit 1000000
```

- Physically recoverable transient data queue:

```
sfsadmin create clusteredfile regionNamecicstdqphfile \
 2 itqkey byteArray 12 itqdata varLenByteArray 32768 \
 cicstdqphidx -unique 1 itqkey sfs_ shortName \
 -preallocate 10 -recordlimit 1000000
```

- Nonrecoverable transient data queue:

```
sfsadmin create clusteredfile regionNamecicstdqnofile \
 2 itqkey byteArray 12 itqdata varLenByteArray 32768 \
 cicstdqnoidx -unique 1 itqkey sfs_ shortName \
 -preallocate 10 -recordlimit 1000000
```

- Nonrecoverable temporary storage queue:

```
sfsadmin create clusteredfile regionNamecicsnrectsqfil \
 2 tsqkey byteArray 11 tsqdata varLenByteArray 32768 \
 cicsnrectsqidx -unique 1 tsqkey sfs_ shortName \
 -preallocate 10 -recordlimit 1000000
```

- Recoverable temporary storage queue:

```
sfsadmin create clusteredfile regionNamecicsrectsqfile \
 2 tsqkey byteArray 11 tsqdata varLenByteArray 32768 \
 cicsrectsqidx -unique 1 tsqkey sfs_ shortName \
 -preallocate 650 -recordlimit 1000000
```

4. For any file with a protection level that is set to **none**, use the following commands to create the ACL.

```
acl_edit ././cics/sfs/serverId/fileName
sec_acl_edit> m unauthenticated:ADIQRUE
sec_acl_edit> m any_other:ADIQRUE
sec_acl_edit> exit
```

For any file with a protection level that is greater than **none**, use the following commands to create the ACL:

```
acl_edit ././cics/sfs/serverId/fileName
sec_acl_edit> m user:cell_admin:ADIQRUE
sec_acl_edit> m user:cics/regionName:ADIQRUE
sec_acl_edit> m group:cics_admin:ADIQRUE
sec_acl_edit> exit
```

5. On the host that has the region, use the following command to expand the Region Definitions (RD) %S symbol to the short name of the SFS server:

```
cicsupdate -c rd -r regionName DefaultFileServer="serverId" \
 LocalQVol="sfs_shortName" \
 LocalQProtectProtectVol="sfs_shortName" \
 LocalTDQVol="sfs_shortName" \
 PhysicalTDQVol="sfs_shortName" \
 NonRecTDQVol="sfs_shortName" \
 NonRecTSQVol="sfs_shortName" \
 RecTSQVol="sfs_shortName"
```

---

## Starting a region from the command line

**Note: On the AIX platform.** The information in this section does not cover the full function of the AIX **startsrc** command, and is concerned only with its use for CICS startup. For a complete description of the command, see the relevant AIX documentation. This command is included in the AIX Base Operating System (BOS) Runtime.

To start a CICS region, you must be logged in with a user ID that has root privileges.

**Note:** If you cold start the region, the following information is discarded:

- The runtime databases that contain nonpermanent resource definitions.
- Any in-flight transaction states, temporary storage queues, transaction data queues, or START requests that were not cleared when the region was shut down. These are cleared during normal shutdowns.

See “Cold starts and warm starts” on page 198 for more information.

You can use the **startsrc** facility on AIX and the **cicsstart** facility on Solaris and HP-UX to start a CICS region, although the CICS control program is the preferred method (see “Using the CICS control program commands for startup” on page 199). After CICS successfully starts and runs postinitialization programs, CICS allows transaction processing. If the environment variable **CICSTRACE** is set, CICS automatically starts product trace.

**On the AIX platform:** Use the **startsrc -s** option to specify the name of the region, for example:

```
startsrc -s cics.regionName
```

where *regionName* is the name of the region. This starts the region, using the attribute values that are specified in the Region Definitions (RD). You can use the **startsrc -a** option to change RD attribute values that are to be used at startup. The syntax follows:

```
startsrc -s cics.regionName -a "attribute=value"
```

If you want to change more than one attribute value, separate the attribute-value pairs with a space, as follows:

```
startsrc -s cics.regionName -a "attribute=value attribute=value"
```

For example, the following command explicitly specifies a cold start for regionA:

```
startsrc -s cics.regionA -a "StartType=cold"
```

You can have multiple instances of CICS by specifying different region names. CICS prevents the starting up of multiple instances of the same region.

**On Solaris or HP-UX:** Use the **cicsstart** command. The syntax follows:

```
cicsstart [-c] [-a] [-o attributeName=attributeValue ...] [regionName]
```

- The **-c** option forces **cicsstart** to perform a cold start of CICS.
- The **-a** option forces **cicsstart** to perform an automatic start of CICS.
- The **-o attributeName=attributeValue** option is used to override Region Definitions for this region startup and specify an attribute-value pair that is to be used in



the startup. Multiple attribute-value pairs can be specified with the **-o** option. Use double quotation marks (") to delimit values containing spaces.

- *regionName* specifies the name of the region that is to be started. If a region name is not specified, CICS starts the region that is specified by the environment variable CICSREGION.

For example, the following command cold starts the region named regionA:

```
cicsstart -c regionA
```

---

## Shutting down the region from the command line

You can use the **stopsrc** facility on the AIX platform and the **cicsstop** facility on the HP-UX or Solaris platforms to stop a CICS region, although the CICS control program (**cicscp**) is the preferred method.

The following procedure shuts down the region from the command line:

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, "Enabling administrator, developer, and end user access to CICS," on page 11.

2.

- **On the AIX platform:** Use the following command for a normal shutdown:

```
stopsrc -s cics.regionName
```

where *regionName* is the name of the region. A normal shutdown completes all current processing and releases resources before shutting the region down. No new requests for work are accepted by the region.

If required, you can use the **-c** (cancel) or the **-f** (forced) options to shut down the region.

When **-c** option is specified, **stopsrc** sends a SIGTERM request to the region and attempts to stop all processes and release all resources before shutting the region down. The region releases all resources and shuts down. If the region does not shut down within the period that is specified in the subsystem object class for the region, **stopsrc** sends a SIGKILL signal to shut down the region. When this happens, some resources that are allocated by the region are possibly not deallocated. If this is the case, you can run the **cicsnotify** command to release these resources and to delete the region lock file.

Use the following command for this type of shutdown:

```
stopsrc -s cics.regionName -c
```

When **-f** is specified, **stopsrc** releases the resources but does not wait for processes to stop before shutting down the region. As with the **-c** option, some resources allocated by the region are possibly not deallocated. If this is the case, you can run the **cicsnotify** command to release these resources, and to delete the region lock file.

Use the following command for this type of shutdown:

```
stopsrc -s cics.regionName -f
```

The information does not cover all the functions of the AIX **stopsrc** facility, and is concerned only with its use for CICS shutdown. For a complete description of the **stopsrc** facility, see your AIX documentation. The **stopsrc** facility is part of the AIX Base Operating System (BOS) Runtime.

- **On HP-UX or Solaris:** Use the following command to stop a region:

```
cicsstop [-f] [-k] [regionName]
```



where *regionName* specifies the name of the region that is to be stopped. If you do not specify a region name, CICS stops the region that is specified by the CICSREGION environment variable.

If the command is issued without the **-f** or **-k** option, a normal shutdown completes current processing and releases resources before shutting down.

The **-f** option forces a stop; **cicsstop** releases the resources but does not wait for processes to stop.

The **-k** option forces an immediate shutdown of the region. All CICS processes are killed; lock files are removed; shared memory is cleaned up.

The following example shuts down the region named regionA immediately:

```
cicsstop -k regionA
```

Shutdown messages are sent to the `/var/cics_regions/regionName/console.nnnnnnn` file.

A normal shutdown can take a long time to complete. You can require users to end long-running tasks, or you can use CEMT to purge the tasks, and allow shutdown to complete normally.

---

## Shutting down an SFS server by using the **cicssfsshut** command

This procedure uses the **cicssfsshut** command to shut down an SFS server.

**Attention:** Before shutting down SFS server, ensure that all regions that use the SFS server have first been shut down.

1. Log in with a user ID that has root privileges and is set up for an administrator as is described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.
2. Use the following command:

```
cicssfsshut ./cics/sfs/serverId
```

where *serverId* is the base name for this server.

For more information about using this command, see the *TXSeries for Multiplatforms Administration Reference*.

---

## Using the **cicsdestroy** command to delete a region

Use the **cicsdestroy** command to delete all resource definitions that are directly associated with a region. An example follows:

```
cicsdestroy -r regionA
```

This command deletes all resource definitions except those that are defined for the SFS server.

You can also delete SFS files that are used by the region. For more information, see the description of the **-s** option that is given in the *TXSeries for Multiplatforms Administration Reference*.

For information about how to use the **cicscp** command set to delete a region, see “Deleting a region by using the **cicscp** command” on page 19.

---

## Using the `cicssfsdestroy` command to delete an SFS server

Use this procedure to delete an SFS server by using the `cicssfsdestroy` command.

This procedure does not remove the operating system user ID that is created for the SFS server.

1. Log in with a user ID that has root privileges and is set up for an administrator as described in Chapter 2, “Enabling administrator, developer, and end user access to CICS,” on page 11.

2. Use the following command to stop the SFS server:

```
cicssfsshut /./cics/sfs/serverId
```

3. Use the following command to destroy the SFS server:

```
cicssfsdestroy /./cics/sfs/serverId
```

This command does the following:

- Deletes the SFS server definitions from the CICS server database
- Deletes the entry in the operating system mechanism that is used for starting and stopping long-running processes

4. Remove the SFS server directory:

```
/var/cics_servers/SSD/cics/sfs/serverId
```

See the *TXSeries for Multiplatforms Administration Reference* for more information about using this command.



---

## Appendix C. Performing CICS administrative functions as a non-root user

This appendix describes how to enable non-root users to perform CICS administrative tasks, for example:

- Starting a CICS region
- Stopping a CICS region

These tasks can be performed by non-root users on Open Systems platforms only; they do not apply to the Windows platform.

The technique that is described in this appendix uses a freeware tool called **Sudo**. Sudo enables System Administrators to configure a set of commands that can be run by a group of users who have specified privileges. For example, the System Administrator can define a group of non-root users, and authorize that group to run a specified set of CICS commands. This group of users can then execute these CICS commands, without first acquiring the root credentials. You can use this method to authorize users to issue any of the CICS administrative commands.

For security reasons, Sudo does not pass the NLSPATH along with the user environment. CICS should be installed in the default location (`/usr/lpp/cics`) to use this procedure, because `LD_LIBRARY_PATH` is not passed along with other environment settings.

To permit non-root users to perform CICS administration tasks:

1. Download and compile Sudo.
2. Copy the CICS message catalog files (`cics.cat`, `cicssfnt.cat`, `cicsmfnt.cat`) to the system default locations:

**AIX** `/usr/lpp/nls/msg/%L`

**HP-UX**  
`/usr/lib/nls/msg`

**Solaris**  
`/usr/lib/locale/%L/LC_MESSAGES`

3. Configure Sudo for the required set of commands, and specify the users who are authorized to run these commands. You must ensure that only authorized users have the necessary privileges, and that the correct set of commands is being configured for use by non-root users.

**Note:** Do not configure Sudo to permit non-root users to issue the **cicssdt**, **cicsddt**, and **cicsodt** commands because of the escape to shell feature that is available with these commands.

The authorized users can now execute the CICS commands that you specified. For example, to start a CICS region, an authorized user could issue the following command from the command line:

**AIX** `#sudo /usr/lpp/cics/bin/cicscp -v start region region-name`

**Solaris or HP-UX**  
`#sudo /opt/cics/bin/cicsstart options`



---

## Glossary

This section contains definitions of those terms and abbreviations, used in the CICS user information, that relate specifically to CICS.

## Numerics

**3270.** An IBM 3270 Information Display System terminal.

**3270 data stream.** The stream that conveys displayable data characters and nondisplayable control characters between a *host processor* and a terminal, as defined in the *3270 Information Display System: Data Stream Programmer's Reference*.

**3270 device emulation.** Support that allows a *local* or *remote device* on one system to appear as a 3270 device to another system.

## A

**ABEND.** EXEC CICS command that is used to terminate a *task* abnormally. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**abend.** Abnormal end of *task*. An *application* can issue an EXEC CICS ABEND command to terminate a *task* abnormally. Synonymous with *abnormal termination*.

**abend code.** A four-character code that is used to identify an *abnormal termination* (for example, ASRA).

**abnormal termination.** Abnormal end of *task*.

**abort.** A possible outcome of the *two-phase commit* protocol. Synonymous with *backout*.

**absolute path.** A *path* that defines how to reach a *file* from the topmost (root) *directory*.

**absolute time.** (1) In CICS Timer Services (START), a time that is relative to the midnight before the time of issuing of the command (the current time). An absolute time earlier than the current time by less than six hours is interpreted as the current time. An *absolute time* earlier than the current time by more than six hours is interpreted as being relative to the midnight following the current time. For example, if the current time is 15:00 hours, an *absolute time* of 11:00 is interpreted as 15:00, and an *absolute time* of 07:00 is interpreted as 07:00 tomorrow. (2) As returned by an EXEC CICS ASKTIME command and input to an EXEC CICS FORMATTIME command, the number of milliseconds since 00:00 on Jan 1, 1900.

**ACB.** Access control block. In VTAM®, an access method control block that links an application to VTAM.

**access.** The ability to read or *update* a resource. In CICS, *access* to protected resources is controlled by *RSL security* or an *External Security Manager (ESM)*.

**access control block.** See *ACB*.

**access method.** A technique for moving data between *main storage* and input/output devices.

**access program.** A user-provided part of a FEPI application that handles the main communications with application programs in CICS or IMS™ systems.

**access right.** An operating system term defining the permissions that are required to *access* a *file* system object.

**ACID.** A fundamental concept of *transaction* processing systems standing for *atomicity*, *consistency*, *isolation*, and *durability*.

**action.** One of the defined *tasks* that an *application* performs. Every condition and attention identifier has a default *action* that is either to ignore the condition or attention identifier, or, in the case of a condition, to terminate the task abnormally.

**action table.** A table maintained by CICS that contains an *action* for every *condition*, *abnormal termination*, and *attention identifier*. Each program at every logical level has a current Action Table that might be the top of a stack of *action* tables for that program.

**active task.** A CICS *task* that is either queued by the *transaction scheduler*, or is being processed in an *application server*, or has been recovered on restart and is awaiting *transaction* resolution in an application server.

**activity keypoint.** Synonymous with *checkpoint*.

**ADDRESS.** EXEC CICS command that is used to get *access* to CICS storage areas-COMMAREA, CWA, EIB, TCTUA, and TWA. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**address space.** The complete range of addresses that are available to a program.

**advanced program-to-program communication.** See *APPC*.

## Glossary

**after-image.** A record of the contents of a data element after it has been changed. After-images are used for *forward recovery* in SFS servers.

**agent.** In a *two-phase commit* syncpointing sequence, a task that receives syncpoint requests from an *initiator*.

**AID.** Attention identifier. Part of the *data stream* that is sent to the CICS *region* by a terminal, indicating which *PF key* or *PA key* (including ENTER, CLEAR, and so on) caused the data to be sent.

**AIX.** A UNIX-compatible operating system from IBM under which CICS for AIX runs.

**AIXwindows.** (CICS for AIX only) IBM's graphical user interface that is based on the Open Software Foundation's OSF/Motif user interface.

**ALLOCATE.** EXEC CICS command that is used to acquire a conversation to a *remote APPC* logical unit for use by an *APPC* mapped conversation. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**ALT.** Application load table.

**alternate facility.** An intercommunication *session* that a *transaction* obtains by means of an EXEC CICS *ALLOCATE* command. Compare *principal facility*.

**alternate index.** An *index* of alternate keys that provides a *path* for secondary *access* to a *KSDS* or *ESDS* file.

**alternate index base file.** The *file* that is the base or normal route of *file access* in an alternate *index* arrangement.

**alternate key.** A *key* in a *file* record other than the *primary key*. An *alternate key* can consist of several fields within a record (see *segmented key*). Any number of alternate keys can exist in a record and they do not have to be unique.

**American National Standards Institute.** See *ANSI*.

**American Standard Code for Information Interchange.** See *ASCII*.

**ANSI.** American National Standards Institute. An organization that establishes procedures for creating and maintaining voluntary industry standards in the USA.

**APAR.** Authorized program analysis report. A report of a problem that is caused by a suspected defect in a current, unaltered release of a program. (CICS for AIX, CICS for Solaris and IBM uses as the basis to supply a temporary corrective fix to a problem. APAR fixes are called corrective service because they are installed to cure problems that are currently being experienced by an installation.

**API.** Application programming interface. The EXEC CICS programming interface that is supported by CICS for *user application* programs. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*. and the *TXSeries for Multiplatforms Application Programming Reference*.

**APPC.** Advanced program-to-program communication. The SNA protocol boundary of the presentation services layer of the LU6.2 architecture. *APPC* is commonly used as a synonym for *LUTYPE6.2*.

**application.** A program or suite of programs that do work. If an application uses FEPI, it is a FEPI application (also known as a terminal front-end program). In VTAM, *application* means programs that communicate directly using VTAM; in a FEPI environment, this means the back-end systems on one hand, and FEPI on the other.

**application identifier.** See *applid*.

**application load table.** See *ALT*.

**application manager.** A CICS *runtime process* that manages a pool of *application servers*.

**application owning region.** The CICS region, involved in *transaction routing*, where the *transaction* is running.

**application program.** (1) A program written for or by a user that applies to the user's work. In this sense, an *application program* is part or all of the implementation of an *application*. (2) In data communication, a program that is used to connect and communicate with stations in a network, enabling users to perform application-oriented activities. (3) The basic unit executed by a CICS *transaction*.

**application programming interface.** See *API*.

**application server.** A CICS *runtime process* that executes CICS *tasks*. This process also manages the *reprepare* and outcome coordination of an *in-doubt Logical Unit of Work* whenever an *application server* was left in this state in a previous incarnation of the region.

**application unit of work.** A set of actions within an *application* that the designer chooses to regard as an entity. The designer decides how, if at all, an *application* should be subdivided into *application* units of work, and whether any *application unit of work* shall consist of one, or many, (CICS) *Logical Units of Work (LUWs)*. Typically, but not exclusively, an *application unit of work* corresponds to a CICS *transaction*.

**applid.** Application identifier. The name by which a logical unit such as a CICS *region* is known in a network. The CICS *applid* is specified as the *region* name on startup.



**archive.** Data that is needed for the purposes of *recovery*. Archives divide into those pertaining to crash *recovery* and those for media recovery.

**archive region.** A *backup* of features such as the *resource definitions* and *maps* that make up a region. The **cicsdefault**, **cicsimport** and **cicsexport** programs manipulate *archive* regions.

**ASCII.** American Standard Code for Information Interchange. The standard code, which uses a coded character set consisting of 7-bit coded characters (eight bits including parity check) that is used for information interchange among many data processing systems, data communicating systems, and associated equipment. The *ASCII* set consists of control characters and graphic characters.

**ASKTIME.** EXEC CICS command that is used to request the current date and time of day. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**ASRA.** CICS *abnormal termination* in an *application program* that is caused by the program's raising an unexpected *exception* or signal.

**ASRB.** CICS *abnormal termination* in an *application program* that is caused by the program's calling a system call incorrectly or exceeding *file system* space limitations.

**assembler language.** A source language that includes symbolic machine language statements in which a one-to-one correspondence occurs with the instruction formats and data formats of the computer.

**ASSIGN.** EXEC CICS command that is used to get values from outside the program's *local* environment. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**asynchronous.** Describes an *event* that occurs at a time that is unrelated to the time at which another *event* occurs. The two events are mutually asynchronous. The relationship between the times at which they occur is unpredictable.

**asynchronous processing.** A means of distributing the processing of an *application* between systems in an intercommunication environment. The processing in each system is independent of the *session* on which requests are sent and replies are received. No direct correlation can be made between requests and replies and no assumptions can be made about the timing of replies.

**ATI.** Automatic transaction initiation. When data is sent to an intrapartition *destination* (PUT requests from one or more programs) and the number of entries in the *queue* reaches a predefined level (trigger level), it

can be optionally specified that a *transaction* is automatically initiated to process the data in that queue.

Transactions can also be automatically started by means of EXEC CICS *START* requests.

The *Automatic Transaction Initiation (ATI)* facility allows a *user transaction* to be initiated either immediately, or, if a terminal is required, when that terminal has no *task* associated with it.

**atomicity.** A basic property of *transaction* processing systems. This term means that transactions (*Logical Units of Work*) must either complete (be *committed*) or appear as though they never happened (be *aborted*). Any intermediate data modifications that are made by a *transaction* must be able to be undone (*rolled back*). It must never be the case that some of the modifications that are requested by a *transaction* take place while others do not.

**attach security.** This level of *security* applies to inbound communication flows. It determines whether a user ID is used in communication between CICS regions.

**attention identifier.** See *AID*.

**attribute.** (1) A component of a 3270 *datastream*. (2) A named characteristic of a *resource class* similar to a *field* name in a record or a column name in a relational database. Attribute names are passed to the CICS RDO commands (for example, **cicsadd** and **cicsget**) as part of an *attribute-value* pair.

**attribute value.** The value of an *attribute* of a particular *resource* in a *resource class*.

**audit trail.** A manual or computerized means for tracing the transactions affecting the contents of a record.

**authentication.** The process performed by a system in verifying that a user is known to it. CICS assumes that a user is verified on his *local* machine by the operating system. Remote users (for example, from *SNA connections*) are verified by CICS by way of the *CESN* transaction.

**authorization.** The process of checking that an authenticated user is allowed *access* to an object. CICS provides *RSL* checking for CICS objects such as transient data queues and user journals, and provides a *user exit* for an *External Security Manager (ESM)*. .

**authorized program analysis report.** See *APAR*.

**autoinstall.** A method of creating a terminal definition, installing it in an active CICS system at the time the terminal is logged on, and deleting it at logoff.

**autoinstall control program.** A user-replaceable CICS program that is used to select some of the data that is

## Glossary

needed terminals to be installed automatically, notably the CICS terminal identifier and the model name that is to be used in each instance.

**automatic transaction initiation.** See *ATI*.

**autostart.** A CICS startup mechanism that applies to CICS regions and to SFS that are started by way of CICS. If the *region* or server has never been previously started, a *cold start* is performed. If the previous *shutdown* was a normal shutdown, CICS does not need to perform any *recovery* processing. If the previous shutdown was not a normal shutdown, CICS controls the *recovery* of its own resources before allowing *access* to the *region* or server.

**auxiliary temporary storage.** *Temporary storage queues* that are held on disk instead of in main memory.

**auxiliary trace facility.** The CICS auxiliary trace facility allows users to record trace information about CICS activity on a *sequential file* instead of in main storage.

## B

**back-end system.** The CICS or IMS system in which existing applications run. Equivalent to **partner** system. See also *front-end system*.

**back-end transaction.** A *transaction* that is started as a result of a *front-end transaction* initiating *Distributed Transaction Processing (DTP)*.

**backout.** The restoring to a previous state of all or part of a system. The process of removing all the updates against protected resources, such as files, that is performed by an *application program* that either has terminated abnormally or was an *in-flight task* at the time of a system failure. Backout can be done dynamically in the case of an *application* abnormal termination, or during restart in the case of a system failure. See *dynamic transaction backout*.

**backup.** The process of making a copy of a data *file* that can be used if the original *file* is destroyed.

**backward recovery.** The process of restoring *integrity* to databases and other recoverable resources following a failure. Before a change is made to an element of a recoverable resource, such as a *file* record or a *database* segment, a *before-image* of the element is recorded. Before-images are used to reverse the changes that were made by logical units of work that were incomplete when the failure occurred. The recoverable *resource* is thus returned to a known state and processing can continue.

**basic conversation.** Type of *APPC* conversation in which the CICS *application* must add control bytes to *application* data for transmission to the partner. A *basic*

*conversation* uses EXEC GDS commands and is not available in CICS. Also known as *unmapped conversation*.

**basic direct access method.** See *BDAM*.

**basic mapping resource.** See *BMS*.

**batch processing.** Type of data processing in which a number of similar input items are grouped for processing serially with a minimum of operator intervention and no end-user interaction.

**BB.** (1) Begin bracket. (2) Between brackets. In *SNA*, an indicator defining the beginning of a conversation.

**BDAM.** Basic direct access method. An *access method* used to retrieve or *update* particular blocks of a *data set* on a *direct access* device. CICS on Open Systems does not support this *access method*.

**before-image.** A record of the contents of a data element before it is changed. Before images are used to *backout* incomplete or incorrect changes in the *event* of a failure.

**begin bracket.** See *BB*.

**begin-session handler.** A user-provided part of a FEPI application that handles begin-session processing.

**between brackets.** See *BB*.

**binary synchronous communication.** See *BSC*.

**bind.** In System Network Architecture (*SNA*), a request to activate a connection between two *logical units*.

**bind race.** In *SNA*, a condition in which two or more logical units (LUs) send bind requests to each other at the same time.

**bind request.** A request to establish a connection between systems.

**bind-time security.** In LU6.2, the level of *security* that is applied when a request to establish a *session* is received from, or sent to, a *remote* system. Used to verify that the *remote system* is really the system it claims to be. Also known as *session security*. See also *link security* and *attach security*.

**bit.** (1) A binary digit, 0 or 1. (2) The smallest unit of computer information, which has two possible states, usually taken to represent the binary digits 0 or 1.

**BLINK.** One of the options of *BMS* extended highlighting, available with a display that uses the extended data stream. *BLINK* causes a character or character *field* to toggle between normal display and no display.

**blocking call.** A *function* (or subroutine) that waits for some *event* to complete and that by so doing prevents a *thread* from continuing until the *event* occurs.

**BMS.** Basic mapping support. A facility that moves data streams to and from a terminal, providing *device independence* and *format independence* for *application* programs. BMS is an interface between CICS and its *application* programs. It formats input and output display data in response to BMS commands in programs. To do this, it uses device information from the operating system and CICS *resource class* definitions, and formatting information from *maps* that you have prepared for your *application* programs. BMS exists in three versions: minimum, standard, and full function. CICS on Open Systems only supports minimum *function* BMS.

**BMS macros.** Basic mapping support (BMS) macros. See *map definition macro*, *map set definition macro*, and *field definition macro*.

**BMS map definition.** Basic mapping support (BMS) map definition. The use of macros (DFHMSD, DFHMDI, and DFHMDF) to define the size, shape, position, potential content, and properties of *BMS map sets*, *maps*, and *fields* within *maps*.

**BMS maps.** Basic mapping support (BMS) maps. *Maps* telling BMS how to format *field* data for display. They are not needed for text data output. Every BMS *mapping* command names a *map* that contains formatting (mapping) instructions. Each *map* has two forms: *physical map* and *symbolic map*.

**BMS processor.** Basic mapping support (BMS) processor. Generates *physical map* and *logical map* files from BMS source files. See *cicsmap*.

**bottleneck.** A place in the system where contention for a *resource* is affecting performance.

**BPS.** Builder parameter set. Part of a message sent or received in *transaction routing*. An encoding of the parameters that are required for a terminal definition. It consists of a VTAM *bind* image (not used by CICS) existence bits (a bit mask of the parameters present in the BPS), and the terminal definition parameters themselves.

**bracket.** In SNA, one or more chains of request units (RUs) and their responses, which are exchanged between two LU-LU half-sessions and represent a transaction between them. A bracket must be completed before another bracket can start. Examples of brackets are data base inquiries and replies, update transactions, and remote job entry output sequences to work stations. The duration of a bracket is shown by the begin-bracket and end-bracket indicators in the request headers of the first and last requests in the bracket. When a bracket is not in progress, the session is between brackets and is in *contention mode*.

**BSC.** Binary synchronous communication. Data transmission in which *synchronization* of characters is controlled by timing signals that are generated at the sending and receiving stations. Compare with *SDLC*.

**buffer.** An area of processing storage that is used to hold a block of data while it is waiting to be processed or written to an I/O device.

**buffer address.** In 3270 data stream, the address of a location in the character buffer (screen image).

**builder parameter set.** See *BPS*.

**byte.** A sequence of eight adjacent binary digits that are operated on as a unit.

## C

**C.** A programming language that is designed for a wide range of system and commercial *applications*.

**CAGE.** A CICS-private transaction that deletes aged *temporary storage queues*.

**CAIN.** A CICS-private transaction that performs work After Initialization.

**CALF.** A CICS-supplied transaction (Convert And Load File) that uses *function shipping* to convert one *file* format to another. For example it can be used to transfer VSAM files to SFS files.

**CANCEL.** EXEC CICS command that cancels *interval control* requests that were previously issued by a *DELAY* or *START* command. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**capacity planning.** An analysis of processor loading and processor capacity, extending into *memory*, other resources (disk, lines), and timings and response where necessary. A method of translating growth in user demands into requirements for future computing resources. It projects future workload by taking into account the increase in existing *applications* and the introduction of new applications, thus allowing a prediction of performance, and helping in the evaluation of future configurations.

**CARP.** A CICS-private transaction (*Asynchronous Request Purger*) that deletes *local* queues when they have been unable to be sent to the target system for longer than a configurable time.

**CATS.** A CICS-private transaction (*Asynchronous Transaction Sender*) that periodically attempts to send *local* queues of EXEC CICS *START* requests to their target systems.

**CAUT.** A CICS-private transaction that periodically gathers statistics.

## Glossary

**CCFLAGS.** Environment variable that is used to store, compile, and link options for C programs.

**CD.** Communication definitions. CDs define the communications between a CICS *region* and another region. The other *region* can be a CICS region, a *CICS family* region, or any LUTYPE6.2 system that supports mapped conversations.

**CDCN (CICS on Open Systems).** A *CICS-supplied transaction* that turns the IBM Application Debugging Program on and off.

**CEBR.** A *CICS-supplied transaction* that browses *temporary storage queues*.

**CECI.** Command-level interpreter CICS-supplied transaction. Enables CICS commands to be entered, syntax-checked, and executed interactively at a 3270 screen. The interpreter performs a dual role in the operation of a CICS system. For the *application programmer*, it provides a reference to the syntax of the complete CICS command-level *application programming interface*. Most of the commands can be carried through to execution, and the results of execution can be displayed. For the system administrator, it provides a means of *interaction* with the system. The command-level interpreter can be invoked by two CICS-supplied transactions, *CECI* and *CECS*. *CECI* invokes the command level interpreter to enter an EXEC CICS command, checks its syntax, and you modify it as necessary. In most cases, you can also process the command. *CECS* invokes the command-level interpreter to check the syntax of an EXEC CICS command but not to process it.

**CECS.** The *command-level syntax checker CICS-supplied transaction*.

**CEDE.** A *CICS-supplied transaction* that invokes the CICS Execution Diagnostic Facility (EDF) to test your command-level *applications* interactively, without having to supply special program testing procedures.

**CEMT.** A CICS-supplied transaction that invokes all the master terminal functions. These functions include inquiring and changing the value of parameters used by CICS, altering the status of system resources, terminating tasks, and shutting down CICS.

**CESE.** A *CICS-supplied transaction* that signs off a user from the CICS system.

**CESN.** A *CICS-supplied transaction* that enables users to sign on to the CICS system.

**chain.** In SNA, a group of one or more logically linked records (request units) that are transferred over a communication line. The ends of the chain are shown by the first-in-chain and last-in-chain indicators in the request headers of the first and last requests in the chain.

**change direction protocol.** In SNA, a data flow control protocol in which the sending logical unit (LU) stops sending normal-flow requests, signals this fact to the receiving LU using the change-direction indicator (in the request header of the last request of the last chain), and prepares to receive requests.

**character.** A contiguous series of 8 bits (one byte) in storage; the smallest addressable unit of storage.

**CHAT.** A *CICS-private transaction* that handles autoinstalled terminals.

**checkpoint.** Checkpoints are snapshots of the current state of the recoverable data that is being used by a CICS region. Checkpoints are used to capture the state of recoverable data between *backups*. This action minimizes the time that is involved in restarting systems that are using that data, by providing a more recent image of it. Checkpoints are written to the region's *log* with a frequency that is determined by the **CheckpointInterval attribute** in the Region Definition (RD).

**CICS.** Customer Information Control System. An IBM licensed program that enables transactions that are entered at remote terminals, to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

**CICS 3270 Terminal Emulator.** Software interface between a user and a CICS region. Run using the **cicslterm** command (CICS on Open Systems) or the **cicslterm** command (CICS for Windows).

**CICS command.** An operating system executable (either a program or a ) provided by CICS that performs CICS specific operations.

**CICS control program.** See *cicscp*.

**CICS default user ID.** The *default user ID* for the region, as specified with the **DefaultUserId attribute** in the *Region Definitions (RD)*. See *default user ID*.

**CICS family.** Used generally to refer to the complete range of *CICS family* products including, for instance, CICS MVS 2.1, CICS/ESA® 3.1, and CICS OS/2®.

**CICS link user ID.** A locally defined user ID that can be associated with inbound requests.

**CICS logger.** A CICS facility that logs recovery information for use during restart, using the operating system's file system.

**CICS Look At Myself.** See *CLAM*.

**CICS message files.** A set of operating system message catalogs that contain the CICS messages. The main *catalog file* is called *cics.cat*. Messages that are supplied with the *statistics* and *monitoring* sample



programs are in files `cicssfnt.cat` and `cicsmfnt.cat` respectively. Messages from the *Transaction Scheduler* are contained in two further catalogs, `trace_ts.cat` and `tsc_status.cat`. All message catalogs are in locale-specific subdirectories.

**CICS monitoring facility.** The CICS component that is responsible for *monitoring* and producing task-related *statistics* information, such as *task* CPU usage and waits for I/O request units on an individual *task* basis.

**CICS on Open Systems.** A generic term that covers CICS for AIX, CICS for HP-UX, and CICS for Solaris.

**CICS PPC gateway.** See *PPC gateway*.

**CICS region user ID.** The user ID assigned to a CICS *region*.

**CICS-private program.** An operating system executable (either a program or a ) that CICS provides for its own internal use.

**CICS-private transaction.** A CICS *transaction* that CICS provides for its own internal use.

**CICS-supplied example program.** A program, supplied by CICS, that demonstrates how some facility can be used.

**CICS-supplied program.** A program, supplied by CICS, that enables modification of the system without changing any standard user interfaces. These programs can be customized or replaced to suit an installation's requirements.

**CICS-supplied sample program.** A program, supplied by CICS, that is replaceable by the customer.

**CICS-supplied transaction.** A *transaction* that CICS provides for customer use.

**CICS-value data area.** See *CVDA*.

**cicsadd.** A CICS *command* that adds a new CICS *resource* to the *permanent database*.

**cicscp.** CICS control program. A command-line interface for configuring and administering CICS, using default values where necessary, to automate simple configurations.

**cicscvf.** A CICS *command* that converts a *file* that contains DFHCNV macros (known as a *conversion template*) into an internal CICS format. The *file* defines the data conversions that are to take place on *function shipping*.

**cicsdb2conf.** (CICS for AIX, CICS for Solaris and CICS for Windows) A CICS *command* that configures CICS resource definitions for DB2 and configures DB2 to manage CICS queues and files.

**cicsdb2import.** (CICS for AIX, CICS for Solaris and CICS for Windows) A CICS *command* that defines files to a DB2 database.

**cicsddt.** (CICS for AIX, CICS for Solaris and CICS for Windows) A CICS *command* that interfaces with the *DB2 diagnostic tool*.

**cicsdefault.** A CICS *command* that configures a named *region* from the *default archive region*. See *cicsexport* and *cicsimport*.

**cicsdefaultservers.** A CICS *command* that creates the default resource files for the SFS, SFS schemas and PPC Gateway servers.

**cicsdelete.** A CICS *command* that deletes CICS resources from the *permanent database*, and optionally from the *runtime database*.

**cicsdestroy.** A CICS *command* that destroys all information about a named *region*.

**cicsdfmt.** A CICS *command* that formats CICS *dump* files.

**cicsexport.** A CICS *command* that creates an *archive region* from a named *region*. See *cicsdefault* and *cicsimport*.

**cicsget.** A CICS *command* that displays CICS resources from the *permanent database*.

**cicsgetbindingstring.** A CICS *command* that displays the binding string that is used to contact a given CICS region or CICS SFS or PPC Gateway server.

**cicsgpipid.** A CICS *command* that shows the operating system process identity of the main process in a named running CICS region.

**cicsshpterm.** (CICS for HP-UX) A CICS *command* that invokes the CICS 3270 terminal emulator to connect to a CICS region on HP-UX.

**cicsimport.** A CICS *command* that configures a named *region* from an *archive region*. See *cicsdefault*.

**cicsinstall.** A CICS *command* that adds *resource* definitions from the *permanent database* into the *runtime database*.

**cicsivp.** A CICS *command* that drives the *installation verification procedures (IVPs)*.

**cicslterm.** (CICS for Windows only) A CICS *command* that provides 3270 emulation and enables connection to a CICS region.

**cicsmap.** A CICS *command* that generates a *symbolic map* and/or a *physical map* from a source *file* that contains mapset macros (DFHMDf, DFHMSD, and DFHMDI).

## Glossary

**cicsmfmt.** A CICS-supplied sample program that formats monitoring files.

**cicsmigrate.** A CICS command that creates a script to migrate CICS regions, databases and other underlying configurations.

**cicsmigrateclass.** A CICS command that adds new attributes to, and deletes old attributes from, the resource definitions for a given class.

**cicsmkcobol.** A CICS command that builds the CICS COBOL support file.

**cicsnotify.** (CICS on Open Systems only) A CICS command to forcibly deallocate resources that remain allocated when the region is no longer running.

**cicsppcgwy.** (CICS on Open Systems only) A CICS command that starts and initializes a PPC gateway.

**cicsppcgwycreate.** (CICS on Open Systems only) A CICS command that creates a PPC gateway.

**cicsppcgwydestroy.** (CICS on Open Systems only) A CICS command that destroys a PPC gateway.

**cicsppcgwylock.** (CICS on Open Systems only) A CICS command that takes or releases a PPC gateway lock.

**cicsppcgwyshut.** (CICS on Open Systems only) A CICS command that shuts down a PPC gateway.

**cicsrlck.** A CICS command that releases a locked region.

**cicssdt.** A CICS command that interfaces with the SFS diagnostic tool.

**cicssfnt.** A CICS-supplied sample program that formats statistics files.

**cicssf.** A CICS command that starts an SFS.

**cicssfscnf.** A CICS command that configures an SFS.

**cicssfscree.** A CICS command that creates an SFS.

**cicssfdestroy.** A CICS command that destroys all information about an SFS.

**cicssfimport.** A CICS command that creates files and indexes on an SFS from resource descriptions that are held in schema files.

**cicssflock.** A CICS command that releases an SFS lock file.

**cicssfshut.** A CICS command that stops an SFS.

**cicsstart.** (CICS for HP-UX, CICS for Solaris and CICS for Windows) A CICS command that starts a CICS region.

**cicsstop.** (CICS for HP-UX, CICS for Solaris and CICS for Windows) A CICS command that stops a CICS region.

**cicstcl.** A CICS command that translates CICS API commands into statements in the supporting language, and compiles and link-edits the resulting source file into an object that CICS can execute.

**cicstcpnetname.** A CICS command that returns the NETNAME that is used by a CICS region on a TCP/IP connection.

**cicsteld.** A CICS command that acts as a telnet server gateway to a CICS region. Any available telnet client on any machine and operating system can be connected to this server, provided that the client provides 3270 emulation.

**cicslterm.** (CICS on Open Systems only) A CICS command that provides 3270 emulation and enables connection to a CICS region.

**cicstfmt.** A CICS command that formats trace files.

**cicstracefilter.** A CICS command that filters a CICS trace file to select or remove specified trace entries.

**cicstran.** A CICS command that translates CICS API commands into statements in the supporting language.

**cicsupdate.** A CICS command that updates a CICS resource in the permanent database, and optionally installs it into the runtime database.

**cicsupdateclass.** A CICS command that changes selectively the value of an attribute for a given class.

**CLAM.** CICS Look At Myself. A CICS-private transaction that runs periodically to check the consistency of the CICS data structures.

**client/server.** A distributed application design model in which the front-end transaction (the one that initiates the conversation) is called the client and controls the course of the conversation. The server receives a request from the client, processes it, and returns the results.

**clserver.** A server that is provided by CICS and used in CICS migration to access the log data that was previously held by the CICS log server that is no longer used.

**clustered file.** An SFS file type in which records with adjacent key values are physically clustered together. The clustered file organization optimizes sequential access through the primary index to records in the file.

**CMLV.** (CICS for Windows only) A CICS-supplied transaction that browses the console message log, console.nnnnnn.

**COBOL.** Common business-oriented language. An English-like programming language that is designed for business data processing applications.

**COBOPT.** Environment variable that stores compile-and-link options for *COBOL* programs.

**code page.** (1) In computer character representation, a list of values (usually single-byte or double-byte values) and the characters that they represent. To cater for different languages and the character requirements of different types of application, an *interchange code* (such as *EBCDIC*) typically defines several code pages. See *interchange code*. (2) A specification of code points for a graphic character set or in a collection of graphic character sets. Within a code page, each code point can have one specific meaning only.

**cold start.** A CICS startup mechanism that applies to CICS regions and to SFS this has been started through CICS. Cold start of both regions and servers ignores any previous *autostart* that might have occurred, and resets all recoverable information. For a *region*, the recoverable information includes *transient data queue* and *temporary storage queue* read and write pointers, *autoinstall terminal* information, and *local queue* information. For an SFS server, its recoverable information is essentially held on its log file. This file is created and an initial volume for the SFS server is also created.

**COLLECT STATISTICS.** EXEC CICS command that returns, to the invoking application, the current *statistics* for a single named resource, or the global statistics for a named *resource class*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**command.** In CICS, an instruction that is similar in format to a high-level programming language statement. CICS commands invariably *include* the verb EXECUTE (or EXEC). They can be issued by an *application program* to make use of CICS facilities.

**command language translator.** The *cicstran* program that prepares a source *application program* that includes EXEC CICS and optionally EXEC *SQL* commands. The translator program translates the EXEC CICS commands into statements in the language of the *application program*. The translator output can be compiled or assembled in the usual way.

**command-level interface.** The high-level programming interface that uses CICS commands that start with the verb EXEC.

**command-level interpreter.** See *CECI*.

**COMMAREA.** Communication area. An area that is used to pass data between tasks that communicate with a given terminal. The area can also be used to pass data between programs within a task.

**commit.** A term that is associated with *logical units of work (LUWs)*. When an LUW is said to have committed, this indicates that all its actions are permanent.

**committed change.** A change that is not backed out in the *event* of a failure. Changes that are made by an *LUW* are committed when the syncpoint at the end of the LUW is complete.

**common business-oriented language.** See *COBOL*.

**common work area.** See *CWA*.

**communication area.** See *COMMAREA*.

**communication definitions.** See *CD*.

**compiler.** A program that converts a *source program* in a *high-level language* to an object module. Before compilation, a CICS *source program* must be processed by the *command language translator* to convert EXEC CICS commands into the source language of the program. Before execution, the translated source program must be processed by the *compiler* and link editor to produce a load module that CICS can execute.

**compute-bound.** The property of a *transaction* where the elapsed time for its execution is governed by its computational content instead of by its need to perform input or output.

**connection.** A target-node pair in the same pool, between which a session can be established (bound), and which can then be used for communication.

**contention mode.** In data communication, a mode of transmission in which any station can transmit whenever the line is available. This occurs when a session is between brackets. If stations transmit simultaneously, protocols determine who wins the contention.

**conversation.** In FEPI, a sequence of related data transmissions between a FEPI application and a particular back-end system. This is analogous to a CICS APPC conversation, but it is not the same as an IMS conversation, and it is not related to CICS conversational mode. In distributed transaction processing, a sequence of exchanges over a session, delimited by SNA brackets. A dialog between two programs in which each program alternately sends and receives data. A dialog between CICS and a terminal user in which CICS alternately accepts input and responds.

**Customer Information Control System.** See *CICS*.

**CVDA.** CICS-value data area. A CICS-supplied value on INQUIRE and SET commands, specifically those that refer to *resource* status or definition. See the *TXSeries for Multiplatforms Application Programming Guide* for more information.

**CWA.** Common work area. An area within the *task-shared pool* that can be used by *application programs* for user data that needs to be accessed by any *task* in



## Glossary

the system. This area is acquired during system initialization. Contrast with *transaction work area* (TWA).

## D

**daisychain.** In CICS intercommunication, the chain of sessions that results when a *region* requests a *resource* in a *remote* region, but the remote *region* discovers that the *resource* is in a third region and has itself to make a *remote* request.

**data integrity.** The quality of data that exists provided that accidental or malicious destruction, alteration, or loss of data are prevented.

**Data Language I.** See *DL/I*.

**data link protocol.** A set of rules for data communication over a data link in terms of a transmission code, a transmission mode, and control and *recovery* procedures.

**data security.** The protection of data against unauthorized disclosure, transfer, modifications, or destruction, whether accidental or intentional.

**data set.** The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. See *file*.

**data stream.** All information (data and control information) that is transmitted through a data channel in a single read or write operation.

**data table.** A *file* whose records are held in main storage.

**database.** A collection of interrelated or independent data items stored together without redundancy to serve one or more applications.

**datagram.** An unreliable *network* data packet that is independent of all other packets and lacks any guarantees of delivery or sequence.

**datagram protocol.** A connectionless, datagram-based transport protocol, such as UDP; an RPC protocol that runs over a connectionless transport protocol.

**DB2 diagnostic tool.** See *DDT*.

**DBCS.** Double-byte character set. A set of characters in which each character is represented by two bytes. Languages, such as Japanese, which contain more symbols than can be represented by 256 code points require DBCS or *MBCS*.

**DCT.** Destination Control Table. In CICS systems other than CICS on Open Systems, a table that contains an entry for each extrapartition, intrapartition, and indirect transient data *queue* that is used in the system, or in connected CICS systems. Its equivalent in CICS

on Open Systems has the more readily understandable name of *Transient Data Definitions* (TDD).

**DDT.** DB2 diagnostics tool. An interactive interface to DB2. Uses *DDT commands* to manage DB2 resources.

**ddt commands.** Invokes the *DB2 diagnostic tool* to manage DB2 resources.

**deadlock.** (1) Unresolved contention for the use of a resource. (2) An error condition in which processing cannot continue because each of two elements of the process is waiting for an *action* by, or a response from, the other.

**default archive region.** A *backup* of the *resource definitions*, *maps*, and so on, the delivered basic CICS region. The **cicsdefault** program can be used to copy this *region* to a named one.

**Default Function Shipping User Conversion Program.** See *DFHUCNV*.

**default region.** The *region* that will be selected when none is specified by commands that require a *region* name.

**default user ID.** The *default user ID* for the region, as specified with the **DefaultUserId** attribute in the *Region Definitions* (RD). The default user ID is used for tasks that are not associated with a signed-on user ID. It is also used for tasks that are run by a user in two conditions: 1. When an attempt to run *CESN* fails. 2. When *CRT*E is used to route to another system, and *CESF* is used to log off, the *default user ID* is used until *CESN* is used to log back on.

**definite response.** See *DRx response*.

**DELAY.** EXEC CICS command that suspends processing of the issuing *task* for a specified interval of time or until a specified time of day. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**DELETE.** EXEC CICS command that deletes a record from a *file*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**DELETEQ TD.** EXEC CICS command that deletes all the transient data that is associated with a particular *intrapartition transient data queue*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**DELETEQ TS.** EXEC CICS command that deletes all the temporary data that is associated with a particular *temporary storage queue*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**DEQ.** EXEC CICS command that causes a *resource* that is currently *enqueued* on by a *task*, to be released for use by other tasks. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**destination.** A *queue* of data that is used with the CICS transient data facility. See *transient data*.

**destination control table.** See *DCT*.

**detail statistics.** *Statistics* that detail the events of a given item; for example, the total number of requests to read from a particular file.

**device.** The term "device" is used in the context of this document to indicate any CICS communication facility, whether physical (for example, a real terminal or printer), or logical (such as a conversation to another CICS region).

**device independence.** The capability to write *application* programs so that they do not depend on the physical characteristics of devices. *Basic Mapping Support (BMS)* provides a measure of device independence.

**DFHMDL.** See *field definition macro*.

**DFHMDI.** See *map definition macro*.

**DFHMSD.** See *map set definition macro*.

**DFHUCNV.** Default Function Shipping User Conversion Program. The default program to convert data that is transferred between CICS transactions for *function shipping* or *transaction routing* user conversion.

**direct access.** A method for retrieval or storage of a VSAM data record that is independent of the record's location relative to the previously retrieved or stored data. Contrast with *sequential access*.

**direct access storage.** (1) A storage device in which the *access* time is in effect independent of the location of the data. (2) A storage device that provides direct *access* to data.

**directory.** In Open Systems or Windows, a *file* containing the names of a collection of files that belong to it. A *directory file* is itself named in another *directory* file, such that the total collection of files in the system is arranged in a hierarchy of directories.

**dispatch (dispatching).** (1) To allocate time on a processor to jobs or *tasks* that are ready for execution. (2) In CICS, to schedule a *task* for execution. Dispatching is done by the *Transaction Scheduler*.

**dispatching priority.** A number that is assigned to *tasks* by the *Transaction Scheduler*. It is used to determine the sequence in which they are to run.

**distributed.** The term *distributed* means that the programs and computerized sources of information that make up a computing environment can be physically located on different computer systems, while still working together as a single logical unit. Transaction processing systems are easily moved to a *distributed* computing environment, because these systems traditionally involve modifying a centralized source of information by submitting modification or information requests from *remote* terminals.

**distributed program link.** See *DPL*.

**distributed transaction.** A *transaction* that can *update* data in many user processes on many machines.

**distributed transaction processing.** See *DTP*.

**distributed unit of work.** See *DUW*.

**DL/I.** Data Language I. A general-purpose data base control system that runs in a virtual-storage environment under VSE, OS/VS1, or MVS. DL/I is **not** supported by CICS.

**double-byte character set.** See *DBCS*.

**DPL.** Distributed program link. DPLs provide a *Remote Procedure Call (RPC)*-like mechanism by *function shipping* EXEC CICS *LINK* commands. This allows the linked-to program to issue unsupported CICS *function shipping* calls, such as to DB2 and DL/I, and yields *performance* improvements for transactions needing to read many records from *remote* files.

**DRx response.** Definite response. In SNA, a value on the form-of-response-requested field of the request header. The value directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request.

**DTB.** Dynamic transaction backout. The process of canceling changes that have been made by a transaction (logical unit of work) to recoverable resources, following a failure of the transaction. In CICS, dynamic transaction backout automatically occurs for all transactions that abort.

**DTP.** Distributed transaction processing. Type of intercommunication in CICS, in which the processing is *distributed* between transactions that communicate synchronously with one another over intersystem links. DTP enables a CICS *application program* to initiate *transaction* processing in a system that supports LU6.2, and that resides in the same or a different processor.

**dual screen mode.** The mode that allows you to use two separate terminals, or two windows on the same physical terminal. Used, for example, to run CEDF.

## Glossary

**DUMP.** EXEC CICS command that allows a *dump* to be generated. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**dump.** A representation of the contents of selected areas of *main storage* that is used to determine whether a program is functioning as intended and to analyze problems. Dumps can be recorded by CICS either as a consequence of failure detected during CICS execution, or upon explicit request. CICS allows you some flexibility in specifying the content and type of *dump* to be produced.

**dump code.** A predefined name by which a *dump* is known.

**dump directory.** The *dump directory* is a root *path*, which contains more than one sub-*directory* to hold *dump* files that are created in a *region* run. The *dump directory* is taken from the value of the **DumpName** attribute in the Region Definitions (RD).

**dump ID number.** A unique suffix that is assigned by CICS to *dump* files it writes.

**duplicate key.** A *key index field* in an SFS *file* that is the same as another *key index field* in that file. Duplicate *key* values are not allowed in the primary indices for entry-sequenced and relative files. Duplicate *key* detection can be set for *OFDs*.

**durability.** A fundamental concept of *transaction* processing systems. Durability refers to the necessity of ensuring that the effects of a *transaction* (*Logical Unit of Work*) are permanent, once it has been completed.

**DUW.** Distributed unit of work. In a *distributed* process, all processing between two syncpoints that is taken by two or more intercommunicating transactions using a *two-phase commit* protocol. A DUW is a *distributed* LUW.

**dynamic backout.** A process that automatically cancels all activities performed by an *application program* that terminates abnormally.

**dynamic transaction backout.** See *DTB*.

## E

**EB.** End Bracket. In SNA, an indicator defining the end of a conversation.

**EBCDIC.** Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters that is used for information interchange among many data processing systems, data communicating systems, and associated equipment.

**EDF.** Execution diagnostic facility. This enables an *application programmer* to test a command-level application program *online* without making any

modifications to the source program or to the program preparation procedure. The facility intercepts the running of the program periodically and displays information about the program at these points. Also displayed are any screens that the user program sends, so that the programmer can converse with the *application* program during testing, just as a user would do on the production system.

**EIB.** Exec interface block. A control block that is associated with each *task* in a CICS program. The EIB contains information that is useful during the execution of an *application program* (such as the *transaction* identifiers) and information that is helpful when a *dump* is being used to debug a program.

**emergency restart.** Initialization of the system following an *abnormal termination* in which information that is recorded on the *system log* is used to accomplish *transaction backout* functions.

**EMP.** Event monitoring post. Point in the CICS code at which CICS *monitoring* data is collected. Two types of EMP are possible: system-defined EMP, which collects predetermined CICS *monitoring* information and which cannot be relocated, and user-defined EMP, which collects *task monitoring* information through EXEC CICS ENTER commands. See *Monitoring Definitions*.

**End Bracket.** See *EB*.

**end session handler.** A user-provided part of a FEPI application that handles end of conversation and end of session processing.

**end of chain.** See *EOC*.

**end-of-day statistics.** CICS *statistics* that are written to an *extrapartition transient data queue* at the quiesce or termination of a CICS run, or at a user-specified time. End-of-day *statistics* are reset by an *end-of-day statistics* collection.

**end user.** Anyone who is using CICS to do a job, usually by interacting with an *application* program (transaction) by means of a terminal.

**ENDBR.** EXEC CICS command that ends a browse on a *file* on a *local* or *remote* system. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**ENQ.** EXEC CICS command that schedules the use of a *resource* by a *task* (enqueue). For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**enqueued.** The state of a *task* that is scheduled to *update* a physical segment of a *database* when another *task* is currently accessing that segment.

**entry-sequenced file.** A record-oriented SFS *file* in which the records are stored in the sequence in which

they were entered into the file. The *primary index* of an *entry-sequenced file* is based on the entry sequence numbers (ESNs) corresponding to the sequence of record insertion.

**entry-sequenced data set.** See *ESDS*.

**EOC.** End of chain. In *SNA*, an exception condition that occurs when the end-of-chain indicator is set in the request/response unit (RU) that is returned from VTAM.

**ESDS.** Entry-sequenced data set. A *VSAM data set* whose records are physically in the same sequence in which they were put into the data set. It is processed by addressed direct *access* or addressed *sequential access* and has no index. New records are added at the end of the data set. Its SFS equivalent type is called *entry-sequenced file*.

**ESM.** External security manager. A *security manager* that operates outside CICS. A sample ESM is provided with the CICS system, and the ESM can be tailored to suit the requirements of each individual system.

**ETR.** External throughput rate. The amount of useful work that is completed in a unit of time. For example, the number of transactions completed per elapsed second.

**event.** An *event* is a particular *attention identifier*, *abnormal termination*, or *condition*. It consists of the type and, in the cases of attention identifier and condition, the value corresponding to the particular attention identifier or condition.

**event-monitoring post.** See *EMP*.

**exception.** (1) An abnormal condition such as an I/O error encountered when processing a *file* or using any resource. See *exceptional condition*. (2) An error that is raised in a threaded program. It is related to a hardware or software problem, and handled by using the threads *exception* handling mechanism.

**exceptional condition.** The reason for the failure of a CICS command, such as FILENOTFOUND returned when a READ command failed to locate the named file.

**exception trace entry.** An entry that is made to the *internal trace* table and any other active *trace* destinations when CICS detects an *exceptional condition*. It gives information about what was happening at the time the failure occurred and what was being used.

**EXEC.** Key word that is used in CICS command language. Most CICS commands begin with the keywords EXEC CICS.

**EXEC interface.** The high-level programming interface that uses CICS commands beginning with the verb EXEC. Synonymous with *command-level interface* and *application programming interface*.

**EXEC interface block.** See *EIB*.

**execution diagnostic facility.** See *EDF*.

**exit.** An *exit* from and return to the CICS *runtime* at a stated functional point. The user can insert code at these points to enhance the program. See *user exit*.

**expiration time.** The time at which a time-controlled CICS *function* is to be started.

**extended binary-coded decimal interchange code.** See *EBCDIC*.

**extended curses.** (CICS for AIX only) AIX/6000 contains two libraries of routines to support input and output to the terminal screen. These are Curses and Extended Curses. For further information, read the relevant operating-system documentation.

**extended data streams.** An extended *data stream* is used, as part of *Basic Mapping Support (BMS)*, to communicate with a device that supports extended color, highlighting, or programmed symbols.

**extended recovery facility.** See *XRF*.

**external response time.** Elapsed time from pressing the terminal ENTER *key* until the *action* that the terminal user requested is completed, and CICS can start the next entry.

**external security manager.** See *ESM*.

**external throughput rate.** See *ETR*.

**extrapartition transient data queue.** A CICS facility for temporarily saving data in the form of queues, called destinations. Each extrapartition TD *destination* requires a *resource definition* that links it to an operating system *file* outside the CICS region. Extrapartition destinations are used for data that is either coming from a source outside the region, or being directed from a source within the *region* to a *destination* outside the region. Extrapartition data that is written by CICS is usually intended for subsequent input to non-CICS batch programs. Examples of data that might be written to extrapartition destinations include logging records, statistics, and transaction error messages. Contrast with *intrapartition transient data*.

## F

**fallback.** Removing a level of support or a facility that has been added, to return to the previous level of support.

**FCT.** File control table. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the files that are accessed by *file* control. Its equivalent in CICS on Open Systems is called *File Definitions (FD)*.



## Glossary

**FD.** File definitions. File definitions identify and describe files for use within a CICS region. The files can be used by application programs and by CICS itself.

**FE.** (CICS for AIX, CICS for Solaris and CICS for Windows) field engineering.

**FEPI.** Front-End Programming Interface. A programming interface of CICS for Windows that enables communication with non-LU6.2 partners by simulating an LU0 device. FEPI allows CICS to communicate with existing applications on LU0 systems without changes to those applications. For details of FEPI, see the *TXSeries for Multiplatforms Front-End Programming Interface for Windows* book.

**FFDC.** First failure data capture. Data relevant to a potentially severe software or hardware problem that is recorded as soon as possible after the condition has been detected. See *symptom records*.

**field.** (1) In a *Basic Mapping Support (BMS)* map, a *field* is a designated area of the map, used to represent a particular piece of data. Programmers can give each *field* a name, and can refer to this name in *application* programs, so that the programs can send data to and receive data from the field, without knowing its location within the map. (2) An individually addressable subdivision of an SFS record containing a specific portion of the data in the record. For example, a record of data about an employee might be subdivided into fields that contain the employee's name, identification number, and salary.

**field definition macro.** The *DFHMDf* macro defines a *field* within a *map* that is defined by the previous *DFHMDI* macro. The *DFHMDf* macro specifies initial attributes that are to be given to fields within a *map*.

**field engineering.** See *FE*.

**FILE.** The various EXEC CICS *file control* commands accept an option specifying the *FILE* to which they apply. CICS maps this *FILE* identifier to a particular SFS or DB2 *file* and optional *index* to that file.

**file.** A *file* in this document always refers to a *Structured File Server (SFS)* or *DB2 file* unless explicitly stated otherwise. SFS files emulate CICS/MVS® *VSAM* data sets.

Do not confuse an SFS or DB2 *file* with a CICS *file*, which is accessed using various EXEC CICS *file control* commands. Multiple CICS files can *map* onto one underlying SFS or DB2 *file*.

Files can be operated on using commands such as EXEC CICS SET OPEN, EXEC CICS SET CLOSED, EXEC CICS READ, EXEC CICS WRITE, and EXEC CICS DELETE.

**file control.** Another name for File Services.

**file control table.** See *FCT*.

**file definitions.** See *FD*.

**first failure data capture.** See *FFDC*.

**FMH.** Function Management Header. In SNA one or more headers that are optionally present in the leading request unit (RU) of an RU chain. It allows one session partner in an LU-LU session to send function management information to the other.

**FMH5.** With *APPC*, the FMH5 (also known as the attach FMH), is sent with the *begin bracket (BB)*, which denotes the beginning of a conversation. It contains the information needed to initiate the back-end transaction. See also *begin bracket*.

**format independence.** The ability to send data to a device without being concerned with the format in which CICS displays the data. The same data can appear in different formats on different devices.

**formatted data interface.** In FEPI, a collective name for the keystroke and screen-image interface.

**formatted dump.** This is a *dump* of most of the control blocks and storage areas in a CICS region. When a *dump* is generated, it is in unformatted form. Users can obtain a *formatted dump* by using the *offline utility cicsdfmt*.

**FORMATTIME.** EXEC CICS command used to select the format of date and time. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**forward recovery.** The process of restoring a *backup* copy and bringing this copy up to date by reapplying changes that have been made to the resources since the *backup* was taken.

**fragmentation.** The breaking up of free storage into small areas (by intervening used storage areas). This leads to the reduction in the amount of effective storage that is available for use.

**FREE.** EXEC CICS command that returns an *APPC* mapped *session* to CICS. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**FREEMAIN.** EXEC CICS command that releases main storage. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**Front-End Programming Interface.** See *FEPI*.

**front-end system.** The CICS system in which the FEPI (Front End Programming Interface) runs to provide access to applications that are running on other systems. See also *back-end system*.

**front-end transaction.** In *synchronous transaction-to-transaction* communication, the *transaction* that acquires the *session* to a *remote region* and initiates a *transaction* on that region. A *front-end transaction* can initiate *Distributed Transaction Processing* (DTP). See *back-end transaction*.

**fullword.** A word that is suitable for processing by a given computer. It is usually treated as a unit.

**function.** A programming language construct that is modeled after the mathematical concept. A *function* encapsulates some behavior. It is given some arguments as input, performs some processing, and returns some results. Also known as subprograms and subroutines.

**Function Management Header.** See *FMH*.

**function shipping.** The procedure, transparent to the *application* program, by which CICS accesses resources when those resources are actually held on another CICS system.

**function shipping transformer.** A suite of subroutines that is used in *function shipping* to effect the transformation of the shipped data and results into formats for flowing over the network.

**fuzzy backup.** A facility that enables a *backup* to be taken without having to stop other work from proceeding.

## G

**gateway.** Refers to the *CICS PPC Gateway* that provides connectivity to *SNA*.

**gateway definitions.** See *GD*.

**gateway server definitions.** See *GSD*.

**GD.** Gateway definitions. The Gateway Definitions define the *PPC gateways* to other systems with which this region can communicate.

**GDS.** Generalized data stream. The *data stream* that is used for mapped conversations on *LUTYPE6.2* sessions.

**generalized data stream.** See *GDS*.

**generic key.** A leading portion of a *file* key, containing characters that identify those records that are significant for a particular application.

**GETMAIN.** EXEC CICS command that requests an area of *transaction* storage of a specified size. It can be optionally initialized to a requested bit configuration. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**global user exit.** An event in CICS at which CICS can pass control to a user-supplied program, then resume

control after the user program finishes. The exit is invoked on every occurrence of the event in CICS.

**group.** A means to collect related *resource definitions* together. Each *resource definition* belongs to at most one group, and resources from different *resource classes* can belong to the same group. The *cicsinstall* command allows the *installation* of a group of definitions into the *CICS runtime*.

**GSD.** Gateway server definitions. GSDs define the characteristics required to start and stop the *PPC gateway*.

## H

**halfword.** A contiguous sequence of bits or characters that constitutes half a computer word and can be addressed as a unit.

**HANDLE ABEND.** EXEC CICS command that handles an *abnormal termination* exit. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**HANDLE AID.** EXEC CICS command that specifies the *label* to which control is to be passed when an *attention identifier* (AID) is received from a display device. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**HANDLE CONDITION.** EXEC CICS command that specifies the *action* that is to be taken if a specific *exceptional condition* occurs. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**handler.** A piece of user code that handles an *abnormal termination*, *condition*, or *attention identifier*. It can be a separate program in the case of an *ABEND* handler, or a labeled section of the same program in all cases. CICS supports only handlers that are labels for COBOL.

**heuristic commit/backout.** A condition in which some external agency has intervened in the *two-phase commit* protocol to force part of a *distributed unit of work* (DUW) to either *commit* or *backout* in possible contravention of the outcome that is eventually delivered by the *coordinator*. This usually arises when *network* communications fail after a DUW has reached the *in-doubt* state; in this case the participants should wait for communications to resume, but the system administrator can intervene to *commit* or *backout* the *in-doubt transaction* and thereby free locked resources. See also *heuristic damage*.

**heuristic damage.** If the outcome delivered to a *transaction participant* differs from that provided as part of a *heuristic commit/backout*, *heuristic damage* is said to have occurred. Such damage is reported.

## Glossary

**high-level language.** A programming language, such as COBOL or C, in which each statement is converted by a *compiler* into one or more machine instructions. A *high-level language* is more application-oriented and less machine-oriented than assembler language. CICS *application* programs must be processed by the CICS translator before compilation. The output of the *compiler* is an object module that must be processed by the *linkage editor* to produce a loadable object that can be executed by CICS.

**high-values.** Hexadecimal FF, as used in COBOL.

**hookid.** These are unique identifiers for the operating system *trace* facility. A *hookid* specifies the text and format of *trace* data.

**host computer.** The primary or controlling computer in a data communication system.

**host processor.** The primary or controlling computer in a multiple computer installation.

**HP-UX.** A Hewlett-Packard operating system under which CICS for HP-UX runs.

**hypertext.** An *online* information retrieval data format that allows easy navigation between articles.

## I

**I/O.** Input/output. Pertaining to the movement of data between a processor and a peripheral device, to the functional unit or channel involved, and to the associated data.

**IBM Software Distribution.** See *ISD*.

**ICE.** Interval control element. An element that is created for each EXEC CICS *START* and *triggered transaction* request.

**icon.** A picture or graphical representation of an object on a display screen to which users can point to with a device, such as a mouse, to select a particular operation, or perform a particular action.

**ID.** Identity or identifier.

**identifier.** See *ID*.

**identity.** See *ID*.

**IGNORE CONDITION.** EXEC CICS command that specifies that no *action* is to be taken if a specific *exceptional condition* occurs. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**immediate shutdown.** A *shutdown* of CICS in which *tasks* in progress are not allowed to complete normally and in which CICS must perform *recovery* actions on restart.

**IMS.** Information Management System. A data base/data communication system that can manage complex data bases and terminal networks.

**in-doubt.** A state of a *transaction* (*Logical Unit of Work*) in which it has completed the first phase of the *two-phase commit* protocol and is awaiting instructions as to whether to *commit* or *abort*.

**in-doubt window.** The period between the sending of a syncpoint request to a *remote system* and the receiving of a reply. During this period, the *local system* does not know whether the *remote system* has committed its changes. If processing fails in the *in-doubt* window, *recovery* processing must resolve the status of any work that is in-doubt.

**in-flight.** The state of an *active task* that has started a *Logical Unit of Work* but not reached the first phase of the *two-phase commit* protocol.

**in-flight task.** A *task* that is in progress when a CICS system failure or *immediate shutdown* occurs.

**inbound.** In FEPI and CICS, data that is received by a program from elsewhere. From the point-of-view of the back-end system, this data is outbound or output to a terminal.

**include.** Term that is used in C guides to refer to a *file* or *directory* from which source code can be copied into a C program, at compile time, by using an embedded C *include* command. A C *include file* is equivalent to a COBOL *copybook*. For further information, see the relevant C documentation.

**index.** An ordered collection of entries that sequences and locates *file* records. Each entry consists of a *key* and a pointer.

**indirect destination.** A transient data *destination* that points to another destination, rather than directly to a queue. Indirect destinations allow you to refer to a single real *destination* by more than one name. For example, different message types can be associated with different indirect destinations. Changes to the *indirect destination* definitions can then cause all messages to go to the same real *queue* or to different queues as required.

**Information Management Systems.** See *IMS*.

**initial data.** A type of inbound data that arrives when a new session is bound. This is commonly called a "good morning" message.

**initialization.** Actions that the CICS system performs to construct the environment in the CICS *region* to enable CICS applications to run.

**initiator.** In a *two-phase commit* syncpointing sequence, the *task* that initiates syncpoint activity. See also *agent*.



**input/output.** See *I/O*.

**installation.** (1) A particular computing system, in terms of the work it does and the people who manage it, operate it, apply it to problems, service it, and use the work it produces. (2) The *task* of loading a software product from its distribution medium onto the computer where it is to run.

**installation verification procedure.** See *IVP*.

**integrity.** See *data integrity*.

**interactive.** Pertaining to an *application* in which each entry elicits a response from a system or program, as in an inquiry system or an airline reservation system. An *interactive* system can also be conversational, implying a continuous *dialog* between the user and the system.

**interchange code.** An accepted convention for computer character representation. An interchange code typically defines several code pages. *EBCDIC* and *ASCII* are interchange codes. See *code page*.

**intercommunication facilities.** A generic term covering *intersystem communication* (ISC).

**internal response time.** Elapsed time between when the message to start a *transaction* is received by CICS, and the end of the transaction.

**internal security.** CICS provides its own validation mechanism to support both transaction-level and resource-level *security* validation. It also allows an alternative validation mechanism to be installed and used instead. A module implementing such a mechanism is called an *External Security Manager* (ESM).

**internal throughput rate.** See *ITR*.

**internal trace.** CICS *trace* service that stores trace entries in a table held in main memory. The table is cyclic, meaning that when the limit of table entries is reached, new *trace* entries overwrite existing ones.

**International Organization for Standardization.** See *ISO*.

**intersystem communication.** See *ISC*.

**interval control.** See *Timer Services*.

**interval control element.** See *ICE*.

**interval statistics.** CICS *statistics* that are gathered at user-specified intervals, then reset. See also *end-of-day statistics*, *requested statistics*, and *unsolicited statistics*.

**intrapartition.** See *IP*.

**intrapartition transient data.** See *TD*.

**IP.** Intrapartition. Data that is internal to a region.

**ISC.** Intersystem communication. Communication between separate systems by means of *Systems Network Architecture* (SNA) and *Transmission Control Protocol/Internet Protocol* (TCP/IP) networking facilities. ISC *links* CICS systems, and can be used for *user application* to user application communication, or for transparently running CICS functions on a *remote* CICS region.

**ISD.** IBM Software Distribution.

**ISO.** International Organization for Standardization. An international body that promotes the development of standards that facilitate international exchange of goods and services, and that encourages cooperation in technological activity.

**ISO model.** International Organization for Standardization (ISO) model. A set of rules for data communication, sanctioned by the International Organization for Standardization (ISO). The ISO protocols enable systems supplied by different vendors to connect and communicate. They are the basis of the open systems interconnection (OSI) standards.

**isolation.** A fundamental concept of *transaction* processing systems. Isolation refers to the necessity of preventing other users from accessing resources acquired by a *transaction* (*Logical Unit of Work*) until that *transaction* commits or aborts.

**ITR.** Internal throughput rate. The number of completed transactions per processor busy second. (Processor busy seconds can be calculated by multiplying elapsed seconds by the processor utilization percentage.)

**IVP.** Installation verification procedure. A program or programs that are run at the end of *installation* of a product to check whether the product is working correctly.

## J

**JCT.** Journal control table. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the journals that are accessed by *journal* control. Its equivalent in CICS is called *Journal Definitions* (JD) .

**JD.** Journal definitions. Journal definitions control destinations for journaling output from a CICS region.

**JOURNAL.** EXEC CICS command that creates a *journal* record. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**journal.** A set of special-purpose sequential files that can contain any and all data that the user needs to facilitate subsequent reconstruction of events or data changes.

## Glossary

**journal control.** See *Journal Services*.

**journal control table.** See *JCT*.

**journal definitions.** See *JD*.

**journal output synchronization.** Ensures that control is not returned to an *application program* that is writing to a *journal*, until the physical write is complete. Implemented by specifying the WAIT option on an EXEC CICS JOURNAL command.

**journal records.** A unit of data that is written by an EXEC CICS JOURNAL command.

**Journal Services.** The CICS service that supports the use of journals.

**journaling.** The recording of information onto any *journal* (including the *system log*), for possible subsequent processing by the user. The primary purpose of *journaling* is to enable *forward recovery* of resources for which this facility is not automatically provided. A *resource* can be reconstructed by applying transactions in the *journal* against a previous version of it. Journaling can also be used for any other user-defined purpose, such as auditing, accounting, or performance analysis.

## K

**key.** (ISO) One or more characters within a set of data that contains information about that set, including its identification.

**Key Sequenced Data Set.** See *KSDS*.

**keypoint.** Synonymous with *checkpoint*.

**keystroke interface.** The part of the Front-End Programming Interface that allows a front-end application to specify a sequence of keystroke-like commands, which are used to define input to a back-end application.

**keyword.** (1) A symbol that identifies a parameter. (2) A part of a command operand that consists of a specific character string.

**KSDS.** Key Sequenced Data Set. A *VSAM file* whose records are loaded in *key* sequence and controlled by an index. Equivalent in SFS to a *clustered file*.

## L

**label.** See *paragraph name*.

**last in chain.** See *LIC*.

**LD.** Listener definitions. LDs define methods that a CICS region should use to listen for requests.

**library.** In Open Systems or Windows, a *file* containing objects that can be linked together to form executables.

**LIC.** Last in chain. See *chain*.

**line.** (1) On a terminal, one or more characters that are entered before a return to the first printing or screen position. (2) A physical connection on a telecommunications network.

**LINK.** CICS command that transfers control from one *application program* to another at the next logical level. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**link security.** A limit on one system's *authorization* to attach transactions and *access* resources in another. See also *bind-time security*.

**link user ID.** A locally defined user ID that can be associated with inbound requests.

**linkage editor.** A computer program that creates one executable object from one or more objects (usually found in a *library*), by resolving cross references among them. All CICS *application* programs need to be processed by the *linkage editor* (link-edited) before execution. See *source program* and *compiler*.

**listener definitions.** See *LD*.

**local.** Pertaining either to resources that are residing on the computer where the CICS *region* is running, or to resources that are directly accessed from the *region* even if they are not located on that processor. For example, a region's transient data queues might be located on an SFS server that is on a separate processor from the region, but these are still viewed as being *local* to the region. Contrast this with *remote*.

**local device.** A device, such as a terminal, that is directly attached to a computer's data channel. Contrast with *remote* device.

**local name.** A name that is meaningful and usable only from within the *cell* where the entry exists. The *local name* is a shortened form of a *global name*. Local names begin with the prefix */.* and do not contain a *cell* name.

**local resource.** In CICS intercommunication, a *resource* that is owned by the *local* system.

**local system.** In CICS intercommunication, the CICS system from whose point of view intercommunication is being discussed.

**locale.** C standard *library* functions that determine the way in which items such as dates and times are formatted in C programs. Locales reflect *local* languages and conventions. A program can select one or more locales for different purposes. A CICS *application program* can run in two locales; the *locale* of the user

who invoked the *transaction* and the *locale* with which the *region* was started. The user sees screens and messages in his own locale, while messages to system queues are in the *locale* that is chosen by the system administrator.

**log.** (1) A recording of changes made to a file. This recording can be used for subsequent *recovery* of the file. See *system log*. (2) An alternative term for the CICS-supplied message queues CSMT and console.

**log force.** A *log force* is an *action* that causes all pending log records to be written to permanent storage. A *log force* is usually associated with committing a transaction, and ensures that the log records that are associated with that *transaction* are actually present in the log. Until a *log force* is done, these records might be stored only in memory, and might therefore be vulnerable to system failures.

**log force groups.** A logical association of the *log* records that are associated with different processes. Grouping the *log* records associated with all processes that are spawned by a specific *transaction* provides a convenient way to ensure that all of the *log* records associated with any processes acting on behalf of a specific *transaction* can be referred to with a single expression or operation.

**log record.** A predefined structure into which the *log* data is formatted. Records have a specific size and format, and contain a particular set of related information. A *log record* can be identified by its log sequence number (LSN).

**logger.** See *CICS logger*.

**logging.** The recording by CICS of *recovery* information onto the *system log*, for use during restart. These records are required for *recovery* and *backout* support by CICS following an *abnormal termination*.

**logical end-of-task.** End of logical unit of work (LUW). Contrast with *physical end-of-task*.

**logical map.** A *logical map* is a source language data structure that the *compiler* uses to resolve *source program* references to fields in a Basic Mapping Support (BMS) map.

**logical unit.** See *LU*.

**logical unit of work.** See *LUW*.

**logical unit of work identifier.** See *LUWID*.

**logical volume.** A logical means of addressing physical data storage. A *logical volume* appears to be physically contiguous storage, though it may actually be composed of regions of disk space on multiple physical disks. A *logical volume* manages a physical volume by controlling all its I/O, and the mounting and dismounting of physical volumes. A logical volume,

while representing regions of disk space on possibly different disks, appears to be a physically contiguous address space.

**logically recoverable.** A *logically recoverable* transient data *queue* is one whose contents CICS can restore, after an error, to the status that they were in at the last LUW.

**logon.** The act of establishing a *session* with the operating system. Contrast with *signon*.

**long running mirror.** A *mirror task* that waits for the next syncpoint in a session, although it logically does not need to do so.

**low-values.** Hexadecimal 00, as used in COBOL.

**LU.** Logical unit. In *SNA*, a port through which a user gains *access* to the services of a network.

**LU-LU session.** Logical unit-logical unit (LU-LU) session. A *session* between two logical units in a *SNA network*.

**LU0.** LUTYPE0. A type of logical unit that is used for communicating with non-SNA terminals, using *binary synchronous communication (BSC)*. LU0 enables the transmission of non-SNA protocols across an SNA network, and is heavily used for connecting non-SNA terminals. LU0 is more primitive than LU6.2. Supported only on CICS for Windows.

**LU1.** LUTYPE1. A type of logical unit that is used for communicating with printers, printer keyboards, and *remote files*. The *file facilities* are more primitive than LUTYPE6.2. Not supported in CICS on Open Systems.

**LU2.** LUTYPE2. A type of logical unit that is used for communicating with 3270 displays. Not supported in CICS on Open Systems.

**LU3.** LUTYPE3. A type of logical unit that is used for sending data to 3270 printers. Not supported in CICS on Open Systems.

**LU4.** LUTYPE4. A type of logical unit that is used for communicating with office systems terminals. Not supported in CICS on Open Systems.

**LU6.** LUTYPE6. A type of logical unit that is used for processor-to-processor communication. LUTYPE6 defines a number of processes.

**LU6.1.** LUTYPE6.1. A type of logical unit that is used for processor-to-processor sessions. LUTYPE6.1 is a development of LUTYPE6. CICS-IMS intercommunication uses LUTYPE6.1 sessions. Not supported in CICS on Open Systems.

**LU6.2.** LUTYPE6.2. A type of logical unit that is used for CICS intersystem (ISC) sessions. LUTYPE6.2 is a development of LUTYPE6.1. The LUTYPE6.2 architecture supports both CICS host to system-level products and CICS host to device-level products. CICS

## Glossary

ISC uses LUTYPE6.2 sessions. *APPC* is the protocol boundary of the LU6.2 architecture. This is the logical unit supported in CICS.

**LUTYPE0.** See *LU0*.

**LUTYPE1.** See *LU1*.

**LUTYPE2.** See *LU2*.

**LUTYPE3.** See *LU3*.

**LUTYPE4.** See *LU4*.

**LUTYPE6.** See *LU6*.

**LUTYPE6.1.** See *LU6.1*.

**LUTYPE6.2.** See *LU6.2*.

**LUW.** Logical unit of work. A sequence of processing actions (for example, *database* changes) that must be completed before any of the individual actions can be regarded as *committed*. When changes are committed (by successful completion of the LUW and recording of the syncpoint on the *system log*), they do not need to be backed out after a subsequent error within the *task* or *region*. The end of an LUW is marked in a *transaction* by a syncpoint that is issued either by the user program, or by CICS, at the end of task. If no user syncpoints exist, the whole *task* is an LUW.

**LUWID.** Logical unit of work identifier (ID). LUWIDs are used to correlate the different parts of a *distributed* transaction.

## M

**macro.** (1) An instruction that, when executed, causes the execution of a predefined sequence of instructions in the source language in which the *macro* is embedded. The predefined sequence can be modified by parameters in the macro. (2) A term borrowed from other *CICS family* members to describe the *DFHCNV*, *DFHMDF*, *DFHMDI*, and *DFHMSD* facilities (which are in fact MVS assembler macros).

**main storage.** Program-addressable storage that is addressable by CICS. This storage can be physical RAM or it can be memory that has been swapped to disk.

**map.** In *BMS*, a format that is established for a screen (printout) or a portion of a screen (printout), or a set of screen format (printout) descriptions. A *map* relates program variables to the positions in which their values appear on a display device. A *map* contains other formatting information such as *field* attributes. A *map* describes constant fields and their position on the display, the format of input and output fields, the attributes of constant and variable fields, and the symbolic names of variable fields.

**map definition.** Definition of the size, shape, position, potential content, and properties of *BMS map* sets, maps, and fields within maps, by means of macros. See *map definition macro*, *map definition macro*, and *field definition macro*.

**map definition macro.** The *DFHMDI macro* defines a *map* within the *map set* that is defined by the previous *DFHMSD* macro. A *map* contains zero or more fields. This *macro* is similar in form to *DFHMSD*. It allows users to override some of the options inherited from *DFHMSD*, and to specify some new ones.

**map set.** One or more *maps* combined into a set. The effects of this combination are to reduce the number of entries in the Program Definitions (PD), and to load simultaneously all maps that are needed for one application.

**map set definition macro.** The *DFHMSD macro* is used to define a set of *BMS maps*. The *macro* consists of a series of operands that define characteristics of the maps in the set. Users can override some characteristics that are specified in *DFHMSD* for individual maps or fields, by specifying operands in the *map* (*DFHMDI*) and *field* (*DFHMDF*) macros.

**map set suffix.** In *BMS* a suffix that relates different versions of a *map set* to different terminal models. This allows you to format the same data differently on different screen types, in response to the same programming request.

**map source file.** The assembler macros that provide a symbolic description of screen layout independent of the actual display terminal.

**mapped conversation.** A form of *APPC* conversation in which the data that is passed to and received from the *APPC application* programming interface is simply user data. The user is not concerned with the internal data formats that the architecture demands.

**mapping.** In *BMS* the process of transforming *field* data to and from its displayable form.

**master terminal.** CICS on Open Systems and CICS for Windows have no concept of a master terminal. In other *CICS* products, it is the terminal at which a designated operator is signed on.

**master terminal functions.** Functions that are provided by the *CEMT* transaction, allowing dynamic control of the system. Invoking these functions, an administrator can (1) change the status and values of parameters that CICS uses, and thereby change the operation of the system; (2) temporarily disable entries in several CICS tables in the *runtime*; (3) terminate any CICS *task* that is currently in the system.

**MCT.** Monitor control table. In other CICS systems, a table that contains the characteristics of the monitoring



that is required. Its equivalent in CICS on Open Systems is called *Monitoring Definitions (MD)*.

**MD.** Monitoring definitions. The MD controls the destination and contents of monitoring output from the CICS region.

**MDT.** Modified data tag. (1) An indicator, associated with each input or output/input *field* in a displayed record, that is set on when data is keyed into the field. The MDT is maintained by the display device and can be used by the program that is using the file. (2) In 3270, a bit in each input *field* that, when set, causes that *field* to be transferred to the host region.

**memory pool.** CICS manages several *real storage* areas. These are the *region pool*, *task-private pool*, and *task-shared pool*.

**message area.** BMS: The area of a screen that is used to send instruction messages to assist the operator in processing a transaction. This area should be separate from the *application* data area to allow communication with the operator, without disturbing the *application* data. The *message area* is normally the bottom one or two lines of the screen.

**metacharacter.** Refers to the following special characters; dollar (\$), at sign (@), period (.), slash (/), dash (-), underscore (\_), percent (%), ampersand (&), question mark (?), exclamation (!), vertical bar or pipe (|), double quote ("), comma (,), semicolon (;), less than (<), greater than (>), and decimal 170.

**metadata.** Data that defines data. Metadata is not data that is supplied by a user, but is information about the structure of user data.

**minimum function BMS.** BMS: Minimum function basic mapping support (BMS). The function, supported by CICS on Open Systems, is a discrete component of the full version and the standard version. *BMS* exists in three versions: minimum, standard, and full. Each version provides a different level of function.

**mirror task.** CICS *task* that services incoming requests that specify a CICS *mirror transaction* (CSMI, CSM1, CSM2, CSM3, CSM5, CPMI, CVMI).

**mirror transaction.** *CICS-private transaction* that recreates a request that is *function* shipped from one system to another, issues the request on the second system, and passes the acquired data back to the first system.

**modified data tag.** See *MDT*.

**monitor.** A user-provided program that handles unexpected events reported by FEPL.

**monitor control table.** See *MCT*.

**monitoring.** (1) The regular assessment of an ongoing production system against defined thresholds to check whether the system is operating correctly. This is in contrast to *performance analysis*, which is the use of one or more *performance* tools (for example, auxiliary trace) to investigate the reasons for *performance* deterioration. (2) Running a hardware or software tool to measure the *performance* characteristics of a system. (3) The facility that provides *performance* data about each CICS *task*.

**monitoring definitions.** See *MD*.

**Motif.** The Open Software Foundation (OSF) graphical user interface.

**mouse.** A hand-held locator that a user operates by moving it on a flat surface. It allows the user to select objects, and scroll the display screen, by pressing buttons.

**MRO.** CICS multiregion operation (CICS Mainframes only).

**multiple virtual storage.** See *MVS*.

**multiple virtual storage/enterprise system architecture.** See *MVS/ESA™*.

**multiprogramming.** The concurrent execution of two or more computer programs by a computer.

**multitasking.** Concurrent running of *application programs* within a CICS region.

**multithreading.** (1) In other *CICS family* products, the use, by several transactions, of a single copy of an *application* program. (2) In CICS on Open Systems the exploitation of the operating system facility that enables one process to handle multiple threads of execution, where each *thread* has its own instruction counter (program counter) and stack memory, with all threads sharing process memory. CICS uses this facility.

**mutex.** A *mutual exclusion mechanism* that is intended for short-term use with internal data structures (as opposed to recoverable data) within an application. Mutexes are thread-dependent because if one *thread* holds a mutex, no other *thread* can *access* the protected data.

**mutual exclusion mechanism.** A means for preventing two separately executing pieces of code from interfering with each other's use of a particular data object. For example, if one *thread* is executing a *function* that modifies a shared data structure, the *application* might need to prevent other threads from attempting to read the data simultaneously, before the modifications are complete.

**MVS.** Multiple virtual storage. An IBM mainframe operating system under which CICS/MVS runs.

## Glossary

**MVS/ESA.** Multiple virtual storage/enterprise systems architecture. An IBM mainframe operating system under which CICS/ESA runs.

## N

**National Language Support.** See *NLS*.

**network.** (1) An interconnected *group* of nodes. (2) The assembly of equipment through which connections are made between data stations.

**network protocol.** A communications protocol from the Network Layer of the OSI *network* architecture, such as the Internet Protocol (IP).

**NLS.** National Language Support. A *feature* that enables the user to communicate with the system in a national language that the user has chosen.

**no response.** SNA: A value in the form-of-response-requested field of the request header (RH) indicating that no response is to be returned to the request, whether or not the request is received and processed successfully.

**node.** (1) In a network, a point at which one or more functional units connect channels or data circuits. (2) SNA: An endpoint of a link or junction that is common to two or more links in a network.

**non-recoverable.** (1) A *resource* that, in the *event* of a *transaction's* being subject to *abnormal termination* or of a system crash, is not restored to the state it was in at the start of the *Logical Unit of Work*. (2) One of the three types of *intrapartition transient data queue*.

**nonconversational.** A mode of CICS operation in which resources are allocated, used, and released immediately on completion of the *task*.

**nonresponse mode.** IMS: A mode of terminal operation that allows asynchronous operations between the terminal operator and the application program. Contrast with **response mode**.

**Null.** A character encoding of Hexadecimal 00 - Low-values in COBOL.

## O

**OFD.** Open file descriptor. Similar to an operating system *file* descriptor, an OFD is a *parameter* that describes the way in which an SFS *file* will be opened, the way in which the file will be used, and the extent to which others can share the data in the file.

**OLTP.** Online transaction processing.

**online.** (1) Pertaining to a user's ability to interact with a computer. (2) Pertaining to a user's *access* to a computer from a terminal.

**online transaction processing.** See *OLTP*.

**open file descriptor.** See *OFD*.

**open system.** A system that implements specified standards that are common across different computer vendors.

**operating system.** Software that controls the running of programs; an *operating system* can provide services such as *resource* allocation, scheduling, input/output control, and data management.

**OS/2.** An IBM operating system under which CICS OS/2 runs.

**outage.** A failure of the CICS system, or planned down time for maintenance or upgrade.

**outbound.** In FEPI and CICS, data that is sent by a program to somewhere else. From the point-of-view of the back-end system, this data is inbound or input from a terminal.

**owner.** In Open Systems or Windows, the identification of the user who creates a *resource* such as a file. CICS is the *owner* of most of the resources it uses.

## P

**PA Key.** Programmed attention key. On a terminal, a *key* that perform various functions as selected by the user or by an *application* program.

**packed decimal.** A data format in which each *byte* in a field, except the rightmost byte, represents two numeric digits. The rightmost *byte* contains one digit and the sign. For example, the decimal value +123 is represented as two bytes with the hexadecimal value X'12 3F' represented in binary as 0001 0010 0011 1111. Also known as internal decimal.

**panel.** A term that is used in the Screen Design Aid. It is equivalent to a *BMS map*.

**panel group.** A term that is used in the Screen Design Aid. It is equivalent to a *BMS map* set.

**paragraph name.** COBOL term for the *destination* of a branch or GOTO instruction (a label).

**parameter.** (1) ISO: A variable that is given a constant value for a specified *application* and that might denote the application. (2) Data that is passed between programs or presented to a program at startup (for example CICS *region initialization* parameters).

**participant.** An *application* is a *participant* in a *transaction* when it either initiates the *transaction* or receives a request on behalf of that transaction.

**partner.** In CICS intercommunication, a transaction that is communicating with a remote transaction or system. In FEPI, this is equivalent to **back-end system**.

**partner LU.** Partner logical unit (LU). In SNA one of an LU pair between which a *session* is established.

**password.** A unique string of characters that a program, computer operator, or user must supply to meet *security* requirements before gaining *access* to data. The *password* is confidential, as opposed to the user identifier which is public.

**path.** In Open Systems or Windows, a description of how to reach a particular *file* by naming the directory hierarchy that must be traversed to reach it. An *absolute path* name provides information from the root of the *directory tree*; a relative *path* name provides information from the current directory.

**pathlength.** The number of instructions executed, for example, per *task*.

**PD.** Program definitions. Program definitions identify application programs, mapsets, and tables to CICS, and define their attributes (such as the source language).

**peer-to-peer.** A form of *distributed* processing, in which the front-end and back-end of a conversation switch control between themselves. It is communication between equals.

**performance.** Together with ease of use, one of the two major factors on which the total productivity of a system depends. Performance is largely determined by a combination of three factors: availability, response time, and throughput. This includes factors such as the internal *transaction* rate, the processor utilization, and virtual storage usage. The study of the *performance* of a system involves making planned measurements, collecting specific data, then drawing educated conclusions about the state of health, or comparative value, of the system.

**performance analysis.** The use of one or more *performance* tools (for example, auxiliary trace) to investigate the reasons for *performance* deterioration.

**performance evaluation.** Determining how well a specific system is meeting, or might be expected to meet, specific processing requirements at specific interfaces. This process does not have as its objective the improvement of the *performance* of a system. However, *performance* evaluation, by determining such factors as *throughput* rate, turnaround time, and constrained resources, can provide important inputs and data for the *performance* improvement process.

**performance improvement.** The increase of the average *throughput rate* and operational capability, or the reduction of turnaround time, or both.

**permanent database.** The collection of *stanza files* that make up the region's *metadata*. One *stanza file* exists per *resource class*. The permanent database is the database that is changed by way of the RDO commands, and is used when the *region* is next *cold started*. Individual and groups of *resource definitions* can be installed into the *runtime* database.

**permission.** The modes of *access* to a protected object. SFSs protect *access* to their resources by using an *access control list*. The operating system controls *access* to its objects by using read, write and execute permissions that are associated with the *owner* of the object, the *group owner* of the object, and the general public.

**PF Key.** Programmed function key. On a terminal, a *key* that perform various functions as selected by the user or by an *application* program.

**physical end-of-task.** End of task. Compare with *logical end-of-task*.

**physical map.** A set of instructions telling BMS how to format a display for a given device. BMS does this by embedding control characters in the data stream.

**physically recoverable.** A *physically recoverable* transient data *queue* is one in which, after an error, CICS can restore the status of its contents to the queue's status as it was at the time of the error.

**PIP.** Program initialization parameters. Data that is formatted in accordance with SNA LU6.2 protocols, and passed to a *back-end transaction* by the CONNECT PROCESS command.

**playing back.** This term refers to the *actions* of CICS when it is restarted. It plays back *log* records for transactions that have *prepared* but that were not actually *committed*.

**PLT.** Program list table. In CICS, a list of programs that are to be executed at a specific phase of system initialization or termination.

**PLTPI.** CICS: Program list table post initialization.

**PLU.** Primary logical unit (PLU). In a SNA LU-LU session, the logical unit that issued the *bind request* that established the session. The PLU contains the primary half-session. The same logical unit can be the PLU in some sessions and the *secondary logical unit* (SLU) in others. In a FEPI environment, the back-end systems are the PLUs.

**pointer.** In computer graphics, the device that is attached to the cursor and tracked on the screen.

**polling.** The process whereby stations are invited, one at a time, to transmit. The *polling* process usually involves the sequential interrogation of several data stations.



## Glossary

**pool.** In the CICS/ESA Front End Programming Interface (FEPI), a collection of **nodes** and **targets**.

**PPC gateway.** The component that allows CICS applications, that are running on a TCP/IP system, to talk to applications that are running in an SNA network using the LU 6.2 protocol.

**prepare.** The first phase of the *two-phase commit* procedure.

**presentation space.** A portion of the device's buffer storage, allocated to a partition, that contains only display data that CICS sends to that partition.

**primary index.** In VSAM and SFS, the set of *primary keys* that provide the standard *path* for *access* to the file.

**primary key.** In each record of a *KSDS*, an identifying *field* or *fields*. The *key* of each record is a *field* in a predefined position within the record. Each *key* must be unique in the file.

**primary logical unit.** See *PLU*.

**principal facility.** The *principal facility* of a *transaction* is the terminal or logical unit that caused a *transaction* to start. For a *back-end transaction* the *principal facility* is a conversation to the *front end transaction*. Compare *alternate facility*.

**priority.** A rank assigned to a *task* that determines its precedence in receiving system resources.

**problem determination.** Starting with a set of symptoms of a problem with a CICS system and tracking them back to their cause.

**process.** An execution of a program. (Similar to the difference between a *transaction* and a *task*.)

**processor.** In a computer, a functional unit that interprets and performs instructions.

**Product Definitions.** See *XAD*.

**product trace.** See *trace*.

**program definitions.** See *PD*.

**program initialization parameters.** See *PIP*.

**program list table.** See *PLT*.

**program list table post initialization.** See *PLTPI*.

**program temporary fix.** See *PTF*.

**programmable terminal.** A user terminal that has processing capability.

**programmable workstation.** See *PWS*.

**programmed attention key.** See *PA key*.

**programmed function key.** See *PF key*.

**property set.** In the CICS/ESA front end programming interface (FEPI), the definition of the characteristics of a pool.

**protected field.** A *field* on a screen that cannot be tabbed to and cannot have text entered into it.

**pseudoconversational.** A type of CICS *application* design that appears to the user as a continuous conversation, but that consists internally of multiple *tasks*. This is also called transaction-oriented programming.

**PTF.** Program temporary fix. A temporary solution or bypass of a problem that is diagnosed by IBM *field engineering* as the result of a defect in a current, unchanged release of the program. See *APAR*.

**purge.** The abending of a *task* to alleviate a short-on-storage condition.

**PWS.** Programmable workstation. A personal computer or terminal that has some *local* processing capabilities.

## Q

**queue.** A *line* or list formed by items in a system that is waiting for service; for example, tasks to be performed, or messages to be transmitted in a message-switching system. In CICS, the transient data and temporary storage facilities store data in queues. See *temporary storage* and *transient data*.

**quiescing.** The process of shutting down the CICS *region* by the rejection of new requests for work.

## R

**RACE.** Resource Access Control Facility. A licensed program, provided by IBM, that provides for access control by identifying and by verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

**random file access.** Location of a record that matches a specific *index key* value. Random *access* of the records in a *file* requires that the file have a unique *index* and that each record have a unique *index key* value.

**RBA.** Relative byte address. The displacement in bytes of a stored record or control interval from the beginning of the storage space that is allocated to the *VSAM data set* to which it belongs.

**RCA.** Region control area. This is the area of memory that CICS uses to store global data.

**RD.** Region definitions. RDs contains both static and dynamic information that is required by CICS at start-up; that is both control and *initialization* information.

**RDBMS.** Relational Database Management System.

**RDO.** Resource definition online. A facility that is used to issue the commands that create and change the *permanent database* and *runtime database*.

**real storage.** The *main storage* in a *virtual storage system*. Physically, *real storage* and *main storage* are identical. Conceptually, however, *real storage* represents only part of the range of addresses that are available to the user of a virtual storage system.

**record identifier.** See *RID*.

**recoverability.** The ability of a *region* to continue processing without loss of data when an unplanned interruption occurs.

**recoverable resources.** Items whose *integrity* CICS maintains in the *event* of a system error. These *include* individual CICS files, auxiliary temporary storage queues, and intrapartition transient data queues.

**recovery.** The process of returning the system to a state from which operation can be resumed. The restoration of resources following an error.

**reentrant.** The *attribute* of a program or *routine* that allows the same copy of the program or *routine* to be used concurrently by two or more *tasks*.

**region.** A named collection of resources that is controlled by CICS as a unit. The collection includes programs, *BMS map sets*, transactions, terminals, files, transient data queues, temporary storage queues, journals, products, and users. One *installation* of CICS can run several regions on the one processor. Regions are likely to be application-specific, but one clear distinction will be between a production *region* and a test region.

**region control area.** See *RCA*.

**region database.** See *permanent database*.

**region definitions.** See *RD*.

**region pool.** Area of *main storage* that is used by CICS *runtime* support code. Each CICS *region* has exactly one region pool. This storage cannot be used by *application* programs.

**relational database.** A *database* that is accessed and organized according to the relationships between data items. Relationships are expressed by means of tables that allow the accessing of items with matching attributes. The *access path* is determined at the time of access. *SQL* is a *relational database manager*.

**Relational Database Management System.** See *RDBMS*.

**relative byte address.** See *RBA*.

**relative record data set.** See *RRDS*.

**relative record number.** See *RRN*.

**relative slot number.** See *RSN*.

**relay transaction.** In *transaction routing*, a CICS-supplied *transaction* that handles communication between a locally-connected terminal and a *transaction* in a *remote* system. The *relay transaction* invokes the relay program.

**reliability.** A measurement of the ability of a system to continue processing without failure. Shutting down an online system to process batch updates to the *database* reduces its availability to end users but has no bearing on the *reliability* of components that are required to deliver the *online* service.

**remote.** (1) In data communication, pertaining to devices that are connected to a data processing system through a data link. (2) Used in CICS for resources that are not directly owned by the *local* region.

**remote ATI.** Remote automatic transaction initiation. A request to run an *asynchronous transaction* on a *remote* region. See *ATI*.

**remote device.** A device, such as a terminal, that is connected to a data processing system through a data link. Contrast with *local device*.

**remote procedure call.** See *RPC*.

**remote resource.** In CICS intercommunication, a *resource* that is owned by a *remote* system.

**remote system.** In CICS intercommunication, a system that the *local* CICS system accesses by using InterSystem Communication (ISC).

**reprepare.** A *function* that is used as part of the *two-phase commit* with presumed *abort* protocol. On restart, reprepare acts on any *in-doubt Logical Units of Work*, using data that is written to the system journal, and re-applies the updates in the data to *commit* or back out the LUWs.

**request header.** See *RH*.

**request/response unit.** See *RU*.

**requested statistics.** These are *statistics* that are requested on a one-off basis. They provide *statistics* from when the *region* started.

**resource.** Any facility of the computing system or *operating system* that is required by a job or *task*,

## Glossary

including main storage, input/output devices, the processing unit, files, and control or processing programs.

**Resource Access Control Facility.** See *RACF*.

**resource class.** A classification of similar resources, such as terminals, transactions, programs, users, files, journals, transient data queues, and temporary storage queues.

**resource definition.** An individual entry in a *resource class*.

**resource definition online.** See *RDO*.

**resource manager.** See *RM*.

**resource protection.** The system *function* of locking on a *resource* to provide exclusive control of that *resource* to a transaction, until the end of a *Logical Unit of Work (LUW)*.

**resource security level.** See *RSL*.

**resource security level checking.** See *RSLC*.

**resource types.** Each CICS *resource* is of a particular resource type (for example, *file* or *transaction*). A CICS *region* has a definition file for each resource type, and these describe the resources that are used by that region.

**response.** In SNA, a message that acknowledges receipt of a request.

**response mode.** A mode in which a system can communicate with an end-user. In IMS, a mode of terminal operation that synchronizes operations between the terminal operator and the application program. Contrast with non-response mode.

**response time.** The elapsed time from entry of a last input message segment to the first response segment. Different viewpoints of activity give a different feeling for response time. Response time externally (message leaving the user to message coming back to the user) is quite different from the internal life of a CICS *task*, although the two are closely related. Task life is one of the major contributors to response time. The *response time* is the elapsed time between pressing the terminal ENTER key and the completion of the *action* that is requested by the terminal user so that the next entry can start.

**restart file.** A *file* that CICS uses to store specific non-critical information that is needed on the next restart.

**resynchronization.** The process by which two *logical units* that are involved in a synclevel syncpoint conversation can determine the outcome of a *transaction* that entered *commit* processing, and was unsuccessful before *commit* processing completed (*in-doubt*).

**RETAIN®.** Database that is used by IBM Support Centers to record all known problems with IBM licensed products.

**return code.** A value that is returned by a program to indicate the result of its processing.

**reverse video.** A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background. For example, changing a red character on a black background to a black character on a red background.

**RH.** Request header. In SNA, control information preceding a request unit.

**RID.** Record identifier. CICS supports a number of different *file* types, each of which permits records to be identified in different ways; for example, by record number or by *key* value. Therefore the identification of a record or records to CICS commands requires different types of data, depending upon the type of *file* that is being accessed. The behavior of CICS commands can frequently be described in generic terms relative to the set of data that identifies a record or records. This identifying set of data is called the Record Identifier (RID).

**RISC System/6000®.** A range of IBM workstations.

**RM.** Resource manager. A server, program or *library* that manages data such as *database* and files. CICS is both a *resource* manager, managing transient data queues, files and temporary storage queues, and a *transaction manager*.

**rollback.** A programmed return to an earlier *checkpoint*. In CICS, the cancellation by an *application program* of the changes made to all *recoverable resources* during the current *Logical Unit of Work (LUW)*. See *backout*.

**routine.** A program or sequence of instructions called by a program. Typically, a *routine* has a general purpose and is frequently used. CICS and the programming languages use routines.

**routing session.** The CICS Supplied Transaction CRTE creates a *routing session* to a *remote* region. Commands that are entered on the *local region* run on a *remote* region transparently to the user. When the CANCEL keyword is entered, the *routing session* is terminated. (A *routing session* is different from a SNA session, which is a connection between two SNA systems.)

**routing transaction.** The CICS Supplied Transaction (CRTE) that enables a user who is at a terminal owned by one CICS *region* to sign onto another CICS region.

**RPC.** Remote procedure call. RPCs are analogous to the standard procedure calls that are used in modern programming, providing modularity and reducing code size. The main difference between standard procedure

calls and *remote* procedure calls is that in a *remote* procedure call, the called procedure is being executed by a different process from the process that is executing the caller. To the calling procedure, a *remote* procedure call looks the same as a call to a *local* procedure. When a program makes a *remote* procedure call to a remote application, the procedure's parameters are automatically bundled into a request message, which is then sent to the *remote* program or system.

**RPC listener.** Remote procedure call (RPC) listener. A server that captures RPC calls from clients. These calls can contain requests to run transactions or kill tasks. The requests are passed to *application servers* through the scheduler function.

**RRDS.** Relative record data set. The displacement in bytes of a stored record or control interval from the beginning of the storage space that is allocated to the *VSAM data set* to which it belongs.

**RRN .** Relative record number. In a relative record *data set* (RRDS), the number of the slot that is used to hold a record; that is, its displacement (in records) from the beginning of the file.

**RSL.** Resource security level. RSLs control which users can *access* a particular *resource* when running a transaction. For RSL validation purposes, *resource* includes transactions that are specified as parameters to EXEC CICS calls. See *security keys*.

**RSLC.** Resource security level checking. The type of checking for *security* of transactions and resources such as files and transient data destinations. See *security keys*.

**RSN.** Relative slot number. The number of the slot that is occupied by a record in an SFS relative file. The RSN is part of the user's data record. Each record in a relative *file* contains a *field* that holds its RSN. See also *RRN*.

**RU.** Request unit. In *SNA*, the basic unit of information that is entering or leaving the transmission subsystem. It can contain data, acknowledgments, control commands, or responses to commands.

**runaway task.** A *task* that has been dispatched and does not return control to CICS. The program that is being used by this *task* is in a loop between two CICS requests.

**runtime.** The collection of processes and data in a *region* that cooperate to execute transactions.

## S

**sample program.** An *application program* that is shipped with the CICS system.

**SCD.** Schema File Definitions. These contain the information that is needed to create files and indexes on an SFS or DB2 database.

**Scheduler Function.** A service within the *region* that is used by other CICS components, such as the *RPC Listener*, to pass *transaction* requests to *application servers*.

**schema.** The data that is used to create files and indexes on an SFS or on a DB2 database.

**schema file.** An operating system *stanza file* containing the data that is used to create files and indexes on an SFS or DB2 database.

**Schema File Definitions.** See *SCD*.

**screen definition facility.** See *SDF*.

**screen-image interface.** The part of the Front End Programming Interface that has a buffer with one byte for each screen position.

**SDF.** (CICS for AIX and CICS for Solaris) Screen definition facility.

**SDLC.** Synchronous Data Link Control. A communications protocol.

**SDT.** Structured File Server (SFS) diagnostic tool. An interactive interface to the SFS. Uses *SDT commands* to manage SFS resources.

**sdt commands.** Used to invoke the *SFS diagnostic tool* to manage SFS resources.

**secondary index.** Any *index* that is used to provide a *path for access* to a *file* other than that provided by the *primary keys*. See *alternate index*.

**secondary logical unit.** See *SLU*.

**secure resources.** The resources for which CICS can perform Resource Security Level (RSL) validation are referred to as secure resources. These resources are Files, Transient Data Queues, Temporary Storage Queues, Journals, Programs, and Transactions.

**security.** Prevention of *access* to, or use of, data or programs without authorization.

**security keys.** The CICS *internal security* mechanism associates an **RSLKey** attribute with each secure *resource* in the database, and a **TSLKey** attribute with each transaction. Users and terminals have corresponding **RSLKeyList** and **TSLKeyList** attributes in the form of a list of one or more of the numeric *security keys*. CICS maintains *internal security* by requiring that the *key* that is specified in the **RSLKey** and **TSLKey** attribute is included in the **RSLKeyList** and **TSLKeyList** for the user or terminal that is attempting to *access* a *resource* or run a transaction.



## Glossary

**segmented key.** A *key* that consists of several fields.

**sequential access.** The retrieval or storage of a *file* record in either its physical sequence or its collating sequence relative to the previously retrieved or previously stored record.

**sequential file.** A *file* whose records are organized on the basis of their successive physical positions, such as on magnetic tape.

**service.** The carrying out of effective problem determination, diagnosis, and repair on a data processing system or software product.

**service elements.** The discrete hardware and software products that provide a terminal user with processing ability.

**session.** A logical connection between an *Advanced Program-to-Program Communication (APPC)* program and a device, or between two *APPC* programs. Contrast with *routing session*.

**session security.** In LU6.2, the level of *security* applied when a request to establish a *session* is received from, or sent to, a *remote* system. Used to verify that the *remote system* is really the system it claims to be. See *bind-time security*.

**SET AUTOINSTALL.** EXEC CICS command that changes some of the values that control automatic *installation* of terminals. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET CONNECTION.** EXEC CICS command that changes some of the attributes that define a connection. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET FILE.** EXEC CICS command that changes some of the attributes of a single named file. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET JOURNAL.** EXEC CICS command that changes the setting of a named journal. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET PROGRAM.** EXEC CICS command that changes some of the attributes of a named program definition. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET STATISTICS.** EXEC CICS command that changes some of the values that control the accumulation and recording of CICS *resource* and system statistics. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET TASK.** EXEC CICS command that is used to *purge* a named user *task*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET TDQUEUE.** EXEC CICS command that changes some of the attributes of a transient data queue. The *queue* must not be *remote* or indirect. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET TERMINAL.** EXEC CICS command that changes some of the attributes of a named terminal definition. This command must not be used for *APPC* sessions. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**SET TRANSACTION.** EXEC CICS command that changes some of the attributes of a named transaction. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**set and test sequence number.** See *STSN*.

**setup program.** A user-provided program that defines and inquires about FEPI resources, and performs housekeeping for the sessions.

**SFS.** Structured File Server. A CICS service that provides a record-oriented file system. SFS offers transactional integrity while supporting large numbers of concurrent users and very large files that can span multiple disks.

**shell.** A software interface between a user and the *operating system* of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens, and communicate them to the operating system.

**shell script.** A set of control statements that has been placed in a *file* and can be executed by a *shell*.

**Shift-Out/Shift-In.** See *SO/SI*.

**shippable terminal.** In *transaction routing*, a terminal whose definition can be shipped to another CICS system when the other system requires a *remote* definition of that terminal.

**Short-On-Storage.** See *SOS*.

**shutdown.** Termination of an invocation of CICS. System administrators can request the *shutdown* process themselves, or it can occur as a result of a system error.

**signon.** In CICS, to perform user identification and verification. The CICS user signs on to CICS using a CICS-supplied transaction, *CESN*.

**Signon Table.** See *SNT*.

**SIT.** System Initialization Table. In CICS systems other than CICS on Open Systems, a table that contains the *start-up* characteristics of the region. Its equivalent in CICS on Open Systems is called the *Region Definitions*.

**SLU.** Secondary logical unit. In a SNA session, the logical unit that received the *bind request* that established the session. The same logical unit can be the SLU in some sessions and the primary logical unit (SLU) in others. In a FEPI environment, the FEPI system is the SLU, and the back-end CICS or IMS system is the PLU.

**SLU P.** An LU0 protocol that is defined by IMS as a protocol to communicate between a programmable workstation, such as a 4700, and IMS. IMS is the Primary Logical Unit (PLU) and the workstation is the Secondary Logical Unit (SLU) in the connection.

**SLU2.** An SLU using LU2 protocols; that is, the 3270 datastream protocol.

**SMIT.** System Management Interface Tool. (CICS for AIX only) The SMIT facility presents a menu driven, task-oriented interface to the many commands that are required to manage a system. SMIT presents complex system management tasks in an organized manner and steps users through the desired *task* with the use of menus and dialogs. SMIT has an hierarchical screen structure and helps users to build and run commands.

**SMP/E.** System Modification Program/Extended. An IBM-licensed program that is used to install software (for example, CICS/ESA) and software changes on MVS/ESA.

**SNA.** Systems Network Architecture. A description of logical structures, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The structure of SNA allows the end users to be independent of, and unaffected by, the specific facilities that are used for information exchange.

**snap dump.** A *dump* that can be requested by a *task* at any time during which that *task* is being processed. Also known as snapshot dump.

**SNT.** Signon Table. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the users of the system. Its equivalent in CICS on Open Systems is called *User Definitions*.

**SO/SI.** Shift-Out/Shift-In. In CICS, control characters that delimit double-byte character set (DBCS) characters in a mixed datastream.

**SOS.** Short-On-Storage. The condition in CICS that occurs when requests for storage from the dynamic storage areas exceed available storage.

**source program.** A set of instructions written in a programming language (such as COBOL or C) that

must be translated into computer language (that is, compiled or assembled) before the program can be executed. CICS *application* programs must be processed by the CICS translator before compilation or assembly. Most programs, and all CICS programs, must be link-edited after compilation or assembly.

**SQL.** Structured Query Language. A *relational database* language that is used to *access* and *update* sets of information.

**SRC.** System Resource Controller. (CICS for AIX only) AIX *subsystem* controller that is designed to minimize the need for user intervention. It provides a mechanism to control *subsystem* processes by using a common *command line* and the C interface. CICS uses the operational control component of SRC to define itself as a *subsystem* within AIX and to perform the *start-up* and *shutdown* functions.

**SRL.** System Reference Library.

**SSD.** Structured File Server Definitions. The SSD contain both static and dynamic information that is required by the SFS server at startup. This is both control and *initialization* information.

**stanza file.** The type of *file* that is used for the *permanent region* database is referred to as a stanza file. Stanza files are text files that are made up of stanzas. A stanza is defined as a name followed by a colon (the stanza name), followed by one or more lines of stanza data (assignments to keywords), followed by a blank line. Each stanza contains a single *resource definition*. In the context of CICS, the stanza name is the *database key* for the resource, and each *line* defines an *attribute* (the attribute is the *keyword* which is assigned one or more values). The first stanza in each *file* is known as the default stanza; it contains default data for the stanza file, and is treated in a special fashion by the *database* routines. One *stanza file* exists for each *resource class*. Do not directly edit the operating system *resource stanza file*.

**START.** EXEC CICS command that starts a *task* on a *local* or *remote system* at a specified time. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**start-up.** The operation of starting up CICS by the system administrator.

**STARTBR.** EXEC CICS command that specifies a record in a *file* or a CICS-maintained *data table* on a *local* or *remote system* where you want the browse to start. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**started transaction.** A CICS *transaction* that is initiated by a terminal user can start other transactions by means of a CICS *START* command. A *transaction* started in this way is known as a started transaction.

## Glossary

**startsrc.** AIX command that starts a subsystem, a *group* of subsystems, or a subserver. Used by CICS to start up a region.

**statistics.** System *statistics* are accumulated continually by CICS management programs in CICS system tables during the running of CICS. System *statistics* can be captured and recorded, either on request or automatically at intervals, by any user whose *security* code allows *access* to such information. In addition, system *statistics* are recorded on normal termination of the region.

Users initiate the recording of *requested statistics* or of automatic *statistics* by means of the EXEC CICS SET STATISTICS command with appropriate arguments. CICS writes the *statistics* to a *file* that is specified in the *Region Definitions (RD)*.

**steady state measurement.** A measurement that is taken after the system has had a chance to stabilize following a system *event* that could affect the parameters being measured.

**stopsrc.** A CICS command that stops a CICS region.

**storage violation.** An error that occurs when an *application program* attempts to *access* data outside its *address space*.

**Structured File Server.** See *SFS*.

**Structured File Server Definitions.** See *SSD*.

**Structured File Server diagnostic tool.** See *SDT*.

**Structured Query Language.** See *SQL*.

**STSN.** Set and test sequence number. In SNA, a communication protocol whereby transmissions can be checked.

**STSN handler.** Set and test sequence number (STSN) handler. A user-provided part of a FEPI application that handles STSN requests.

**subroutine.** A sequenced set of instructions that can be used in one or more programs and at one or more points in a program. The execution of a *subroutine* is usually invoked by a call.

**subsystem.** A secondary or subordinate system.

**summary statistics.** Statistics that summarize the events in a given category; for example, the total number of get requests on all files.

**surrogate.** In CICS *transaction routing*, a terminal entry in the *transaction-owning region* that is used to represent the terminal that invoked, or was acquired by, the transaction. See *surrogate terminal*.

**surrogate terminal.** A terminal whose terminal definition is shipped from a *terminal owning region*.

**SUSPEND.** EXEC CICS command that relinquishes control to *tasks* of higher or equal dispatching priority. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**symbolic links.** Links to resource definitions that are shared between regions.

**symbolic map.** Synonymous with *logical map*.

**symptom records.** A message that is generated to report a symptom when CICS detects potential software or system damage. The information in the record is automatically generated.

**synchronization.** In CICS, a coordinated commitment control process between communicating transactions that ensures that all logically-related updates to *recoverable resources* are completed or that all are backed out.

**synchronization level.** The level of *synchronization* (0, 1, or 2) that is established for an *APPC session* between intercommunicating CICS transactions. Level 0 gives no *synchronization* support, level 1 allows the exchange of private *synchronization* requests, and level 2 gives full CICS *synchronization* support with *backout* of all updates to *recoverable resources* if failure occurs. Also known as *sync level*.

**synchronous.** An operation that occurs regularly or predictably with regard to the occurrence of a specified *event* in another process; for example, the calling of an input/output *routine* at a coded location in a program. Contrast with *asynchronous*.

**Synchronous Data Link Control.** See *SDLC*.

**SYNCPOINT.** EXEC CICS command that is used to divide a *task* into smaller logical units of work by establishing syncpoints. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**syncpoint.** A logical point in execution of an *application program* where the changes made to the databases by the program are consistent and complete and can be committed to the database. The output, which has been held up to that point, is sent to its destinations, the input is removed from the message queues, and the *database* updates are made available to other applications. When a program terminates abnormally, CICS *recovery* and restart facilities do not *backout* updates before the last completed syncpoint. See also *logical unit of work*.

**syncpoint agent.** Any *transaction* that receives a syncpoint request (issued by the syncpoint initiator) during a conversation in a dynamic *transaction* processing environment. See *syncpoint initiator*.



**syncpoint initiator.** The *transaction* that initiates syncpoint activity for a *distributed unit of work*. Contrast with *syncpoint agent*.

**system.** In CICS, an assembly of hardware and software that can provide the facilities of CICS for a particular installation.

**system dump.** A *dump* of the whole CICS region.

**System Initialization Table.** See *SIT*.

**system integrity.** The quality of a system that exists provided that accidental or malicious destruction, or alteration of the system are prevented.

**system log.** The log file that CICS uses to *log* all changes that are made to *recoverable resources* for the purpose of *backout* on restart.

**System Management Interface Tool.** See *SMIT*.

**System Modification Program/Extended.** See *SMP/E*.

**system programming interface.** A subset of the CICS application programming interface that accesses special system-oriented CICS services.

**System Reference Library.** See *SRL*.

**System Resource Controller.** See *SRC*.

**system trace.** This is used for investigating CICS system problems. It optionally writes *trace* to the operating system trace facility.

**Systems Network Architecture.** See *SNA*.

## T

**target.** In the CICS/ESA Front End Programming Interface (FEPI), the back-end CICS or IMS system to which FEPI appears as a secondary logical unit.

**task.** In CICS, a single instance of the execution of a transaction. Contrast with *transaction*.

**task-private pool.** Memory that is allocated to each *application server* to satisfy *local* needs; for example, *GETMAIN* requests.

**task-shared pool.** Task-related storage area. This storage is shared across all *tasks* in a region; for example, storage that is allocated to the Common Work Area (CWA), which is accessible by all tasks that are running in a CICS region.

**task control area.** See *TCA*.

**TCA.** Task control area. An area of *main storage* that is acquired by CICS when a *task* is first dispatched. It is used to control the processing of the task. When acquired, the TCA exists until the *task* is terminated. It contains the current status of the task, its relative

dispatching priority, and parameters and information that are being passed between CICS and the *application program*.

**TCP/IP.** Transmission Control Protocol/Internet Protocol. A communications *subsystem* that allows users to set up *local* area and wide area networks.

**TCT.** Terminal Control Table. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the terminals that are accessed by terminal control. Its equivalent in CICS on Open Systems is called *Terminal Definitions (WD)*.

**TCTSE.** Terminal Control Table System Entry. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the *remote* systems that are accessible from this region. Its equivalent in CICS on Open Systems is called *Communications Definitions (CD)*.

**TCTTE.** Terminal control table terminal entry. In the CICS terminal control table (TCT), an entry for each terminal known to CICS.

**TCTUA.** Terminal User Area. Used to pass data between transactions that are associated with a terminal.

**TD.** (1) Transaction Definitions. The Transaction Definitions define all the transactions that can be processed by CICS. It contains control information that is to be used by CICS to identify and initialize a transaction. (2) Transient Data. A CICS facility for temporarily saving data in the form of queues, called destinations. All intrapartition TD destinations are held as queues in three SFS files: one for *logically recoverable*, one for *physically recoverable*, and the other for *non-recoverable*. Data is written to the *queue* by a user *task*. The *queue* can be used subsequently as input data by other tasks within the CICS region. All *access* is sequential, governed by read and write pointers. When a record has been read, it cannot be read subsequently by another task. An intrapartition *destination* requires a *resource definition* that contains information that finds the *queue* in the intrapartition data set. Applications that might use intrapartition queues *include* message switching, data collection, and queuing of orders.

**TDD.** Transient Data Definitions. The TDD contains entries that define each of the symbolic destinations that transient data is routed to and from. The name of every transient data *queue* must be known to CICS before it is initialized, so users cannot create a transient data *queue* with only an arbitrary name, as they can for *temporary storage queues*.

**temporary conversation.** A FEPI conversation that is allocated from the pool that is specified by the POOL option. It exists only for the duration of a FEPI CONVERSE command.

**Temporary Storage.** See *TS*.

## Glossary

**Temporary Storage Definitions.** See *TSD*.

**Temporary Storage Table.** See *TST*.

**terminal.** (1) A point in a system or communication *network* at which data can either enter or leave. (2) In CICS, a device, often equipped with a keyboard and some kind of screen, that can send and receive information over a communication channel.

**terminal control.** See *Terminal Services*.

**Terminal Control Table.** See *TCT*.

**Terminal Control Table System Entry.** See *TCTSE*.

**terminal control table terminal entry.** See *TCTTE*.

**Terminal Definitions.** See *WD*.

**terminal front-end program.** A CICS application that is designed to use the front end programming interface (FEPI) to communicate with existing back-end applications.

**Terminal Input/Output Area.** See *TIOA*.

**terminal operator.** The user of a terminal.

**terminal owning region.** Region in which a terminal that is participating in *transaction routing* is located.

**Terminal Services.** The CICS services that handle data translation, transaction identification, synchronization of input and output operations, and the session control that is needed to read from or write to a terminal.

**Terminal User Area.** See *TCTUA*.

**thread.** A single sequential flow of control within a *process*.

**throughput.** The total data processing work that is successfully completed during an evaluation period.

**throughput rate.** The data processing work that is successfully completed per unit of time. Throughput rate is analogous to power in engineering, and applies to any workload.

**Timer Services.** CICS Timer Services, in conjunction with the time-of-day clock that is maintained by the operating system, provide commands that can be performed at a specific time.

**TIOA.** Terminal Input/Output Area. Area that is set up by storage control and chained to the terminal control table terminal entry (TCTTE) as needed for terminal *input/output* operations.

**TM.** Transaction Manager. A software unit that coordinates the activities of *resource managers* transactionally.

**TPN.** Transaction Program Name. A term used in SNA for a program that processes transactions in an SNA network. The TPN specifies the name of the *remote transaction* program that is to be connected at the other end of a mapped conversation.

**trace.** Facility for recording CICS activity.

**trace entry points.** The points at which *trace* entries are produced during CICS operation. They are of two types: *system trace entry points* and *user trace entry points*.

**trace facility control.** The CICS Trace facility can be selectively enabled or disabled. Trace can be disabled for the whole region, in which case no *trace* whatsoever is output. If the master *trace* flag is enabled, race can be further enabled for system, EXEC interface, master user, and single *tasks*.

**Trace hook-id.** Unique identifiers that are used by the *trace* facility. A hook-id specifies the text and format of *trace* data that follows.

**transaction.** A unit of *application* data processing (consisting of one or more *application* programs) that is initiated by a single request, often from a terminal. A *transaction* might require the initiation of one or more tasks for its execution. Contrast with *task*.

**transaction backout.** The process of removing the effect of *transaction* activity on CICS resources, such as files. The cancelation, as a result of a *transaction* error, of all updates that a task performs. See *backout*.

**transaction deadlock.** A condition in which two or more transactions cannot continue processing because each is waiting on a *resource* that the other holds.

**Transaction Definitions.** See *TD*.

**transaction dump.** A *dump* of the control blocks and storage areas that are associated with a particular *task* or storage area that is requested by the user.

**transaction identifier.** A name of up to four characters that is specified when the *transaction* is defined to CICS, and that is used to invoke the transaction. For example, to select a transaction, a *terminal operator* enters the *transaction* identifier.

**transaction list table.** See *XLT*.

**Transaction Manager.** See *TM*.

**Transaction Program Name.** See *TPN*.

**transaction rate.** The number of units of processing that are successfully completed per unit of time.

**transaction routing.** Transaction routing (part of the intercommunication facilities) is a facility that allows a terminal owned by one CICS *region* to run a *transaction* that resides in another CICS region.

**transaction routing transformers.** Routines that are used in CICS to encode and decode outgoing and incoming messages during *transaction routing*.

**transaction scheduler.** The *transaction scheduler* is a server that channels *transaction* requests from clients onto *application servers* that run the requests.

**Transaction Security Level.** See *TSL*.

**transaction work area.** See *TWA*.

**Transient Data.** See *TD*.

**Transient Data Definitions.** See *TDD*.

**translator.** See *command language translator*.

**Transmission Control Protocol/Internet Protocol.** See *TCP/IP*.

**trcrpt.** (AIX only) This operating system command reads a specified *trace log* file, formats the trace entries, and writes a report to standard output. For further information, see the operating system documentation.

**trigger level.** The number of records that are written to an intrapartition transient data *destination* or *queue* that cause CICS to automatically initiate a *task* to process that queue.

**triggered task.** This is a *transaction* that is triggered by CICS for transient data queues that are defined as having a *trigger level* and a trigger transaction.

**TS.** Temporary Storage. A CICS facility for temporarily saving data in the form of sequential queues. A *TS queue* is held in *main storage* or on one of two SFS files, one for recoverable queues and one for non-recoverable. See *transient data*.

**TSD.** Temporary Storage Definitions. The Temporary Storage Definitions describe the available templates that determine the attributes that are given to temporary storage queues for which CICS is to provide *recovery*. When CICS first creates a temporary storage queue, CICS uses the longest matching template name.

**TSL.** Transaction Level Security specifies those users who are allowed to run any given transaction. See *security keys*.

**TST.** Temporary Storage Table. In CICS systems other than CICS on Open Systems, a table that contains the characteristics of the temporary storage queues. Its equivalent in CICS on Open Systems is called *Temporary Storage Definitions (TSD)*.

**tuning.** The process of adjusting *operating system* and CICS operands to better meet the users *performance* objectives. The efficiency of one *resource* might have to be set against the inefficiency of some other resource.

**turnaround time.** The elapsed time between entry of the first character of the first input into the input interface and the passage of the last character of the last output through the output interface. The total time consumed from the start to the completion of a specific unit of work measured at specific interfaces. When multiple inputs and multiple outputs are parts of one unit of work, intermediate *turnaround time* specifications might be needed.

**TWA.** Transaction work area. An optional extension of the *task control area TCA*, which is used as a work area for a given *task*. The TWA can be used for the accumulation of data and intermediate results during the execution of the task. When the amount of working storage for a *task* is relatively static, the TWA can be used if data is accessed by different programs during *task* processing. This approach cannot be used for multiple transactions; the TWA is released automatically at *task* termination.

**two-phase commit.** A protocol for the coordination of changes to *recoverable resources* when more than one *resource manager* is used by a single transaction. (Examples of a *resource manager* are CICS, SFS, DB2, Informix and Oracle.) In the first phase all resource managers are asked to *prepare* their work; in the second phase they are requested by the *coordinator* either to *commit* or to *backout* the effects of the transaction.

## U

**UD.** User Definitions. Defines all users who are allowed to *access* a CICS region.

**unbind.** In SNA, to deactivate a session between logical units.

**UNLOCK.** EXEC CICS command that releases exclusive control that was established in response to a read command with the UPDATE option. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**unmapped conversation.** Type of *APPC* conversation in which the CICS *application* must add control bytes to application data for transmission to the partner. A *basic conversation* uses EXEC GDS commands; these are not supported by CICS.

**unprotected fields.** A *field* on a screen that can be tabbed to, and into which text can be entered.

**unsolicited data.** A type of inbound data that arrives on a connection where no FEPI conversation is active.

**unsolicited-data handler.** A user-provided part of a FEPI application that handles unsolicited inbound data.

**unsolicited statistics.** CICS *statistics* that are automatically gathered by CICS for a dynamically

## Glossary

allocated and deallocated *resource* (for example, an autoinstalled terminal) when the *resource* is about to be deleted.

**update.** To modify a *file* with current information.

**upward compatibility.** In CICS, the ability of a later release of the product to run a program that was written for an earlier release.

**User Definitions.** See *UD*.

**user exit.** A place in a CICS module at which CICS can transfer control to a program that you have written (a user-exit program), and at which point CICS can resume control when your exit program has finished its work. See *user exit point*.

**user exit point.** A user exit point, normally abbreviated to user exit, is a point in a CICS module at which CICS can transfer control to a user-written program, called a *user exit program*. See *user exit*.

**user exit program.** A user-supplied program that is given control by CICS at a *user exit point*. The program returns control to CICS when it has finished its work.

**user journal.** User *journal* files are special purpose byte-oriented files. Applications use these journals to *log* information to enable the production of *recovery* routines, audit procedures, and reports. *Logging* is performed by using the EXEC CICS *JOURNAL* and EXEC CICS *WAIT JOURNAL* commands. CICS does not provide facilities to read such journals from within a CICS application. Special operating system programs must be written to perform the required processing offline.

**user trace.** Provided to assist *application* programmers to debug and investigate their programs. *User trace* is the trace of an individual *task*, that is written while a *task* is running.

**user ID.** User identifier (ID). A string of characters that uniquely identifies a user to a system.

**utilities.** Informal term for utility programs.

## V

**Virtual Storage Access Method.** See *VSAM*.

**Virtual Storage Extended.** See *VSE*.

**Virtual Telecommunications Access Method.** See *VTAM*.

**VSAM.** Virtual Storage Access Method. An *access method* for direct or sequential processing of fixed- and variable-length records on *direct access* devices. CICS emulates this by using the *Structured File Server* (SFS).

**VSE.** Virtual Storage Extended. An IBM mainframe operating system under which CICS/VSE® runs.

**VTAM.** Virtual Telecommunications Access Method. An *access method* that is used by some *CICS family* members.

## W

**WAIT CONVID.** EXEC CICS command that is used by an *application* to ensure that any accumulated data is transmitted on an *APPC* mapped conversation. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**WAIT JOURNAL.** EXEC CICS command that synchronizes a *task* with the output of one or more *journal records* that have been created but whose output has been deferred. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**WAIT TERMINAL.** EXEC CICS command that ensures that a terminal operation has completed on an LUTYPE6.2 logical unit. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**warm start.** Initialization of the *region* by using selected region status information that was obtained during the previous termination. Same as *autostart*.

**WCC.** Write control character. A control character that follows a write command in the 3270 data stream and provides control information for executing display and printer functions. A character that is used with a write-type command to specify that a particular operation, or combination of operations, is to be performed at a display station or printer.

**WD.** Terminal Definitions. Used to define the terminals (workstations) that are available for use in a CICS region.

**WIMP.** *Window-Icon-Mouse-Pointer* environment. This type of environment presents a graphical interface that allows the use of *access methods* other than a keyboard.

**window.** (1) A rectangular area of the screen that can be moved about, placed on top of, or pulled under, other windows, or iconized by the user. (2) In *curses*, the internal representation of what a portion of the display might look like at some point in time. Windows can be any size, from the whole display screen to a single character.

**Window-Icon-Mouse-Pointer environment.** See *WIMP*.

**Windows.** A Microsoft® operating system under which CICS for Windows runs.



**WRITE.** EXEC CICS command used (in different forms) to write a record either to a *file* on a *local* or a *remote* system, or to a user-maintained data table on a *local* or a *remote* system. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**write control character.** See WCC.

**WRITEQ TD.** EXEC CICS command that writes transient data to a transient data queue. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**WRITEQ TS.** EXEC CICS command that stores temporary storage data in a *temporary storage queue* in main or *auxiliary storage*. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

## X

**X/Open Distributed Transaction Processing XA interface.** See *X/Open XA*.

**X/Open XA.** The *X/Open Distributed Transaction Processing XA* interface. A proposed standard for *distributed transaction* communication. The standard specifies a bidirectional interface between *resource* managers that provide *access* to shared resources within transactions, and between a Transaction Service that monitors and resolves transactions.

**X-Windows.** (AIX Only) A network-transparent windowing system that was developed by Massachusetts Institute of Technology (MIT). It is the basis for AIXWindows, which runs on the AIX operating system.

**XAD.** Product Definitions. CICS Product Definitions are used to hold information that is needed to interface with other transactional products that are using the *X/Open XA* protocol.

**XCTL.** CICS command that transfers control from one *application program* to another at the same logical level. For programming information, see the *TXSeries for Multiplatforms Application Programming Guide*.

**XLT.** Transaction list table. CICS control table that contains a list of transaction identifications. Depending on a system initialization specification that can be changed during system termination, the transactions in a particular XLT can be initiated from terminals during the first quiesce stage of system termination. During CICS execution, the suffix of an XLT can be entered at the master terminal; the transactions in that XLT can then be enabled or disabled as a group.

**XRF.** Extended Recovery Facility. A facility that increases the availability of CICS transaction processing, as seen by the end users. Availability is

improved by having a second CICS system (the alternate system) ready to continue processing the workload, if and when particular failures that disrupt user services occur on the first system (the active system).

## Y

**yielding loop.** A loop that is characterized by returning control at some point to a CICS *routine* that can suspend the looping *task*. However, the looping *task* will eventually be resumed, so the loop will continue.

## Z

**ZERO.** Value used in *DFHMD* macro.



---

## 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 to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS 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 information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.



Licensees of this program who 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 Corporation  
ATTN: Software Licensing  
11 Stanwix Street  
Pittsburgh, PA 15222  
U.S.A.

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 International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks and service marks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States, other countries, or both:

|                                               |                        |
|-----------------------------------------------|------------------------|
| Advanced Peer-to-Peer Networking <sup>®</sup> | AIX                    |
| AS/400 <sup>®</sup>                           | CICS                   |
| CICS/400 <sup>®</sup>                         | CICS/6000 <sup>®</sup> |
| CICS/ESA                                      | CICS/MVS               |
| CICS/VSE                                      | CICSplex <sup>®</sup>  |

|                                   |                         |
|-----------------------------------|-------------------------|
| C-ISAM™                           | Database 2™             |
| DB2                               | DB2 Universal Database™ |
| GDDM®                             | IBM                     |
| IBM Registry™                     | IMS                     |
| Informix                          | Language Environment®   |
| MVS                               | MVS/ESA                 |
| OS/390®                           | OS/2                    |
| OS/400®                           | RACF                    |
| RETAIN                            | RISC System/6000        |
| RS/6000                           | SOM®                    |
| Systems Application Architecture® | System/390®             |
| TXSeries                          | TCS®                    |
| VisualAge®                        | VSE/ESA™                |
| VTAM                              | WebSphere               |
| z/OS®                             |                         |

Domino®, Lotus®, and LotusScript are trademarks or registered trademarks of Lotus Development Corporation in the United States, other countries, or both.

ActiveX, Microsoft, Visual Basic, Visual C++, Visual J++, Visual Studio, Windows, Windows NT®, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Acucorp and ACUCOBOL-GT are registered trademarks of Acucorp, Inc. in the United States, other countries, or both.

Pentium® is a trademark of Intel® Corporation in the United States, other countries, or both.



This software contains RSA encryption code.



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



---

# Index

## Special characters

%H shorthand 8  
%R shorthand 8  
%S shorthand 8  
+ character (CEMT) 122

## Numerics

3270.key file  
specify SOSI 86

## A

A147abend 190  
accessing CICS resources  
outside the running region 100  
ActivateOnStartup attribute  
description 115  
use of 116  
ACUCOBOL-GT  
cicsprACUCOB file 177  
configuring for use with relational  
databases 174  
creating cicsprACUCOB 177  
language method file 177  
relinking the runtime 177  
aixterm model 85  
application, programs and  
monitoring 219  
applications  
EXEC CICS INQUIRE 123  
EXEC CICS SET 123  
managing resources 123  
ATI, security 95  
attributes  
DefaultUserId 272  
User exit number (PD) 186  
attributes, use of  
ActivateOnStartup 116  
CICSPassword (UD) 15  
EAMLoad 91  
EAMModule 91  
ErrorIsolation 233  
ESMLoad (RD) 94, 101  
FileRSLCheck (RD) 101, 107  
InitialTransid (CTG.INI) 81  
IntrospectInterval 237  
JournalRSLCheck (RD) 101, 107  
LoadDataNumBuckets (RD) 227  
MaxServer 237  
MinServer 237  
NetName  
CTG.INI 81  
Port (CTG.INI) 81  
ProgramRSLCheck (RD) 101, 107  
Protocol  
CTG.INI 81  
LD 81  
Protocol (LD) 83  
RSLCheck (TD) 101, 102, 107

attributes, use of (*continued*)  
RSLKey 103  
RSLKeyList (UD) 103  
ServerIdleLimit 237  
StartType (RD) 280  
TaskSHNumBuckets (RD) 227  
TCPAddress (LD) 83  
TCPProcessCount (LD) 83  
TCPService (LD) 83  
TemporaryStorageRSLCheck  
(RD) 107  
TransactionRSLCheck (RD) 101, 107  
TransientDataRSLCheck (RD) 101,  
107  
TSLCheck (RD) 94  
TSLKey (TD) 95  
TSLKeyList (UD) 95  
UpperCaseSecurity (CTG.INI) 81  
authenticating user access to CICS 89  
authentication using CICS 89  
authorizing access to CICS  
overview 99  
resource definition using CICS  
commands 99  
Authorizing access to CICS  
resources from applications 100  
authorizing access to cics objects, using  
security keys 102  
authorizing access to CICS  
transactions 94, 96  
authorizing user access to CICS, startup  
and shutdown programs 99  
autoinstall models 85  
autoinstall, terminal autoinstall user  
program 189  
autoinstalled terminal definitions,  
recovery of 240  
Automatic Task Initiation (ATI) 193  
automatic transaction initiation,  
security 95  
auxiliary storage trace 221  
auxiliary temporary storage queues,  
recovery of 240  
auxiliary temporary storage, performance  
costs 237  
auxiliary trace 221  
availability problems, file managers 201

## B

backout, list of recoverable  
resources 240  
backup  
fuzzy 255  
incremental 255  
media archiving 245  
Media Recovery Archiving  
(MRA) 243  
SFS 242  
SFS logical volumes 242

backup and recovery  
after a complete failure 251  
of CICS subsystem entries 251  
of configuration 249  
of DB2 247  
of Oracle 249  
of subsystem entries 251  
of user IDs 251  
overview 239  
preventing loss of log data 253  
recovering different types of  
service 252  
tolerating hardware faults 253  
using a second processor 254  
using mirrored volumes 254  
Basic Mapping Support (BMS), NLS 128  
browse cache, use of 233  
BufferPoolSize, improving  
performance 228  
building applications  
WebSphere MQ 179

## C

C  
DB2 transaction support 150  
Oracle transaction support 156  
transaction support for DB2 150  
transaction support for Oracle 156  
C transactions, configure Sybase for 163  
cancel (forced abnormal) shutdown 209  
CDS name file manager  
availability problems 201  
startup failures 201  
CEBR, security 103  
CECI, security 103  
CEDF  
performance costs 236  
security 97, 103  
cell\_admin\_pw environment variable, use  
of 199  
CEMT 119  
asterisk character 122  
blank fields in a screen 123  
DISCARD 120  
INQUIRE 120  
overtyping a screen 122  
PERFORM 120  
PF key values 121  
request formats 121  
screen layout 120, 121  
scrolling a screen 122  
SET 120  
CEMT PERFORM SHUTDOWN, use  
of 211  
CEMT transaction  
+ character 122  
tab key 123  
change, Region Definitions (RD) 20  
changing, password 15

- cics
  - group 99
- CICS
  - clock, setting 238
  - cold starting 280
  - post-initialization programs 190
  - shutting down 211
  - starting up 200
  - startup 190, 199, 280
  - tuning performance 223
- CICS authentication 89
- CICS configuration, backup and recovery 249
- CICS control program
  - cicscp create telnet, use of 73
  - cicscp destroy telnet, use of 73
  - region shut down 207
  - shutting down the region 207
  - Telnet server 73
- CICS Control Program
  - cicscp command 9
  - introduction 9
  - startup 199
- CICS log, file system 22
- CICS password, changing 15
- CICS regions
  - adding 273
  - configuring 273
  - shutdown 210
  - starting 280
  - startup 199
- CICS resource management commands, configuring CICS regions 273
- CICS subsystem entries
  - backup and recovery 251
  - recovery 251
- CICS system with data, backup and recovery needed 253
- CICS system without data, backup and recovery needed 253
- CICS user IDs
  - backup and recovery 251
  - recovery 251
- cics\_admin group 138
- CICS\_BROWSE\_CACHE environment variable 233
- CICS\_ESM\_RETURN\_FAIL 110
- CICS\_ESM\_RETURN\_INVALID 110
- CICS\_ESM\_RETURN\_OK 110
- CICS\_SFS\_DATA\_SIZE environment variable, use of 29
- CICS\_SFS\_DATA\_VG environment variable, use of 29
- CICS\_SFS\_LOG\_SIZE environment variable, use of 29
- CICS\_SFS\_LOG\_VG environment variable, use of 29
- CICS\_SFS\_SIZE environment variable, use of 29
- CICS\_SFS\_VG environment variable, use of 29
- CICS-supplied transactions, security 103
- cicscdt commands, create 140
- cicscp
  - create telnet, use of 73
  - destroy telnet, use of 73
  - introduction 9
- cicscp (*continued*)
  - Telnet server 73
- cicscp command 9, 199
- cicscp create telnet 71
- cicscp start all, use of 200
- cicscp start region, use of 199
- cicscp start sfs\_server, use of 199
- cicscp stop all, use of 207, 211
- cicscp stop region, use of 210
- cicscp stop sfs\_server, use of 211
- cicsdb2conf command 63
- CICSDB2CONF\_CONNECT\_USING environment variable, use of 63
- cicsdb2import command 138
- cicsddt commands, create 140
- cicsdefault command 272
- cicsexport, sharing resource definitions 125
- cicsimport, sharing resource definitions 125
- cicslterm
  - permission to use 100
- cicsmap, permission to use 100
- cicsmfmt 221
- cicsmkcobol
  - use of 175
- cicsoraconf command 64
- CICSORAPASSWORD environment variable, use of 65
- CICSPassword attribute (UD), use of 15
- cicsprACUCOB
  - creating 177
- cicsprACUCOB file, ACUCOBOL-GT 177
- cicsprCOBOL
  - placement of 175
- cicsprCOBOL file, Micro Focus Server Express COBOL 175
- CICSREGION environment variable, use of 261
- cicsrl 233
- cicssdt commands, create 139
- cicssfmt headings 217
- cicssfs command 276
- cicssfsconf command 276
- cicssfscreate command 275
- cicssfsdestroy, use of 283
- cicssfsimport command 137
- cicssfsshut, use of 282
- cicsstart (HP or Solaris) command 280
- cicsstop (HP or Solaris) command 281
- cicstcl, permission to use 100
- cicsteld
  - description 71
  - permission to use 100
  - registered with inetd daemon 73
  - running as a foreground process 76
  - server 72
  - use in DBCS or MBCS environment 76
- cicstfmt, trace utility program 221
- CICSTRACE environment variable, use of 280
- cicstran, permission to use 100
- class, sharing 125
- clients
  - configuring for Telnet server 71
- clients (*continued*)
  - configuring in CICS 71
  - listener process 81
- clock
  - CICS 238
  - monitoring facility 219
  - system 238
- cold start
  - installing groups of resources 115
  - using permanent database 115
- Collating language field (AIX only)
  - specify for DBCS/MBCS environment (AIX only) 30
- commands
  - CICS region (shutdown) 210
  - CICS region (starting) 280
  - CICS region (startup) 199
  - cicsadd 66, 68, 133
  - cicscp 9
  - cicsdb2conf 63
  - cicsdb2import 138
  - cicsddt 140
  - cicsdefault 272
  - cicsdelete 66, 68, 133
  - cicsoraconf 64
  - cicssdt 139
  - cicssfs 276
  - cicssfsconf 276
  - cicssfscreate 275
  - cicssfsimport 137
  - cicsstop (HP or Solaris) 281
  - cicsupdate 66, 68, 133
  - region (shutdown) 210
  - region (starting) 280
  - region (startup) 199
  - shutdown 281
  - stopsrc (AIX) 281
- commands, use of
  - CEMT PERFORM SHUTDOWN 211
  - cicscp create telnet 73
  - cicscp destroy telnet 73
  - cicscp start all 200
  - cicscp start region 199
  - cicscp start sfs\_server 199
  - cicscp stop all 211
  - cicscp stop region 210
  - cicscp stop sfs\_server 211
  - cicsmkcobol 175
  - cicssfsdestroy 283
  - cicssfsshut 282
  - cicsstart (HP-UX and Solaris) 280
  - EXEC CICS PERFORM SHUTDOWN 212
  - RESTORE DATABASE 248, 249
  - startsrc (AIX) 280
- Communications Definitions (CD), sharing resource definitions 125
- complete failure, recovery procedure 251
- configuration
  - backup and recovery 249
- configuring
  - CICS clients 71
  - CICS regions with DB2 36
  - CICS regions with Oracle 45
  - DB2 for CICS 34
  - Oracle for CICS 44

- configuring (*continued*)
  - stages of CICS configuration 9
  - using the SMIT tool 10
- console.*nnnnnn*
  - shutdown information 210
  - storage stress messages 226
- core files
  - maximum file size 274
- CPMI, security 103
- create (cicsddt command) 140
- create (cicssdt command) 139
- CSM1, security 103
- CSM1, security 103
- CSTD transaction 225
- CTG.INI attributes, use of
  - InitialTransid 81
  - NetName 81
  - Port 81
  - Protocol 81
  - UpperCaseSecurity 81
- CTG.INI, file 81
- CTG.INI. attributes, use of
  - InitialTransid 81
  - NetName 81
  - Port 81
  - Protocol 81
  - UpperCaseSecurity 81

## D

- data file
  - setup 133
- databases
  - integrating 150, 156
  - security 90
- DB2
  - availability problems 201
  - backup and recovery 247
  - cicsdb2conf command 63
  - cicsdb2import command 138
  - cicsddt command 140
  - configuration procedures 31, 32
  - configuring 138, 140
  - configuring for queue management 63
  - creating for CICS regions 36
  - DLCHKTIME 39
  - enabling the XA interface 150
  - field name restrictions 50
  - file management 31, 32
  - LOCKTIMEOUT 39
  - queue and file management 37
  - queue management 31, 32
  - recovery 247
  - requirements 32
  - runtime error 37
  - Schema File Definitions (SCD) 138
  - security 90
  - security considerations 40
  - startup failures 201
  - table space 39
  - TP\_MON\_NAME 40
  - transaction support, C 150
  - transaction support, COBOL 150
- DB2 (CICS)
  - configuring 34

- default user ID
  - creating 272
- DefaultUserId attribute (RD), use of 272
- development system, backup and recovery needed 253
- DISPLAY environment variable, use of 259
- DLCHKTIME, DB2 39
- DOS client 78, 81
- dump file, printing 210
- dump request user exit 185
- dynamic distributed program link 185
- dynamic transaction backout 208
- dynamic transaction routing 185

## E

- EAM (External Authentication Manager)
  - overview 91
- EAM External Authentication Manager)
  - use of 188
- EAMLoad attribute (RD), use of 91
- EAMModule attribute), use of 91
- ECI applications
  - enabling access 78, 81
  - Listener Definitions (LD) 78, 81
- enabling the RDBM XA interface 150, 156
- end-of-day statistics 214
- environment variables
  - cell\_admin\_pw, use of 199
  - CICS\_BROWSE\_CACHE 233
  - CICS\_SFS\_DATA\_SIZE, use of 29
  - CICS\_SFS\_DATA\_VG, use of 29
  - CICS\_SFS\_LOG\_SIZE, use of 29
  - CICS\_SFS\_LOG\_VG, use of 29
  - CICS\_SFS\_SIZE, use of 29
  - CICS\_SFS\_VG, use of 29
  - CICSDB2CONF\_CONNECT\_ USING, use of 63
  - CICSORAPASSWORD, use of 65
  - CICSREGION, use of 261
  - CICSTRACE, use of 280
  - DISPLAY, use of 259
  - LANG
    - setting 12
    - use of 15, 21
  - LD\_LIBRARY\_PATH (Solaris)
    - use of 111
  - LIBPATH (AIX)
    - use of 111
  - NLSPATH
    - setting 12
    - use of 15, 21
  - PATH, setting 12
  - SHLIB\_PATH (HP-UX)
    - use of 111
  - time zone (operating system) 238
  - WebSphere MQ 179
- environment variables AIX
  - LIBPATH 12
- environment variables HP-UX
  - SHLIB\_PATH 12
- environment variables Solaris
  - LD\_LIBRARY\_PATH 12
- EPI applications
  - enabling access 78, 81

- EPI applications (*continued*)
  - Listener Definitions (LD) 78, 81
- ERZ0070xxE message is logged 201
- ERZ021004E message is logged 201
- ERZ0220xxE message is logged 201
- ERZ0230xxE message is logged 201
- ERZ0250xxE message is logged 201
- ERZ045001I message 104
- ERZ045013E message 110
- ERZ045014E message 110
- ERZ061013E message is logged 201
- ERZ087001E message 110
- ESM (External Security Manager)
  - overview 106
  - use of 188
- ESMLoad attribute (RD), use of 94, 101
- event-monitoring points (EMPs) 219
- EXEC CICS INQUIRE 123
- EXEC CICS PERFORM SHUTDOWN, use of 212
- EXEC CICS SET 123
- EXEC CICS START PROTECT requests, recovery of 240
- exit programs 186
- exporting a region 273
- ExtDS attribute, use of 85
- extended data stream attribute, use of 85
- External Authentication Manager (EAM)
  - overview 91
- external call interface applications
  - enabling access 78, 81
  - Listener Definitions (LD) 78, 81
- external disks, in backup and recovery 254
- external presentation interface applications
  - enabling access 78, 81
  - Listener Definitions (LD) 78, 81
- External Security Manager (ESM)
  - overview 106
  - use of 189
- external trace 221

## F

- FD attributes, use of, RSLKey 103
- field name restrictions, DB2 50
- field name restrictions, Oracle 54
- field names (DB2), restrictions 50
- field names (Oracle), restrictions 54
- file control, performance costs 237
- File Definitions (FD)
  - adding 133
  - changing 133
  - cicsadd 133
  - cicsdelete 133
  - cicsupdate 133
  - configuring 133, 135
  - deleting 133
  - sharing resource definitions 126
- File Definitions (FD) attributes, use of, RSLKey 103
- file management
  - configuration procedures (DB2) 31, 32

- file management (*continued*)
  - configuration procedures (Oracle) 40, 41
  - configuration procedures (SFS) 24
  - configuration procedures (SQL) 31
  - configuring DB2 138, 140
  - configuring File Definitions (FD) 133
  - configuring SFS 137, 139
  - naming of (SQL) 62, 64
  - requirements (DB2) 32
  - requirements (Oracle) 41
  - requirements (SFS) 24
  - Schema File Definitions (SCD) 137, 138
- file manager
  - availability problems 201
  - managing resources 23, 59, 133
  - startup failures 201
- file system, CICS log 22
- FileRSLCheck attribute (RD), use of 101, 107
- files
  - CTG.INI 81
  - security 103
  - setup 133
- forced abnormal shutdown 209
- fuzzy backup 255

## G

- gateway, startup failures 201
- group
  - cics 99
  - cics\_admin 138
  - of resources 115

## H

- HACMP
  - high availability solution 255
- hardware faults, tolerating 253
- hft model 85
- hft-mb model 85
- High Availability solution 255
- HP 9000 Series 800, use in backup and recovery 254
- hp model 85
- hpterm model 85

## I

- IBM CICS Client
  - product autoinstall 188
- IBM CICS Universal Client
  - for DOS 81
  - for Macintosh 81
  - for OS/2 81
  - for Windows 81
  - for Windows 3.1 81
  - for Windows 95 81
- IBM CICS Universal Client for DOS 78
- IBM CICS Universal Client for Macintosh 78
- IBM CICS Universal Client for OS/2 78
- IBM CICS Universal Client for Windows 78

- IBM CICS Universal Client products
  - enabling access 81
  - Listener Definitions (LD) 81
- IBM RS/6000, use in backup and recovery 254
- ibm3151 model 85
- ibm3161 model 85
- ibm3164 model 85
- immediate shutdown 208
- implementing user exits 186
- importing a region 273
- in-doubt condition 132
- incremental backup 255
- InDoubt condition 132
- inetd 71
- inetd daemon 73
- Informix
  - setting up the XA interface 153
  - transaction support, C 153
- InitialTransid attribute (CTG.INI), use of 81
- inquire about
  - resources 120
- installation
  - WebSphere MQ 179
- interval statistics 214, 216
- intrapartition transient data queues, configuring SFS for 68
- intrapartition transient data, recovery of 240
- IntrospectInterval 237

## J

- JD attributes, use of, RSLKey 103
- Journal Definitions (JD)
  - adding 217
  - changing 217
  - cicsadd 217
  - cicsdelete 217
  - cicsupdate 217
  - configuring 217, 218
  - deleting 217
- Journal Definitions (JD) attributes, use of, RSLKey 103
- journal management, configuring Journal Definitions (JD) 217
- JournalRSLCheck attribute (RD), use of 101, 107
- journals
  - security 103
  - setup 217

## L

- LANG environment variable
  - setting 12
  - use of 15, 21
- language method file, ACUCOBOL-GT 177
- language method file, Micro Focus Server Express COBOL 175
- LD
  - configuring 82
  - ECI applications 78, 81
  - EPI applications 78, 81

- LD (*continued*)
  - external call interface
    - applications 78, 81
  - external presentation interface
    - applications 78, 81
  - IBM CICS Client products 78
  - IBM CICS Universal Client products 81
- LD attributes, use of
  - Protocol 81, 83
  - TCPAddress 83
  - TCPProcessCount 83
  - TCPService 83
- LD\_LIBRARY\_PATH
  - environment variables Solaris 12
- LD\_LIBRARY\_PATH (Solaris)
  - environment variable
    - use of 111
- lft model 85
- lft-md model 85
- LIBPATH
  - environment variables AIX 12
- LIBPATH (AIX) environment variable
  - use of 111
- limitations
  - listener process 82
- links, symbolic 125
- Listener Definitions (LD)
  - configuring 82
  - ECI applications 78, 81
  - EPI applications 78, 81
  - external call interface
    - applications 78, 81
  - external presentation interface
    - applications 78, 81
  - IBM CICS Client products 78
  - IBM CICS Universal Client products 81
- Listener Definitions (LD) attributes, use of
  - Protocol 81, 83
  - TCPAddress 83
  - TCPProcessCount 83
  - TCPService 83
- listener process, clients 81
- listener process, limitations 82
- ln command, linking databases 125
- LoadDataNumBuckets attribute (RD), use of 227
- loading, Product Definitions (XAD) 115
- LOCKTIMEOUT, DB2 39
- logical unit of work, overview 239
- logical volumes
  - copying 245
- logical volumes (SFS), creating 274
- logical volumes, enlarging 30
- logically recoverable transient data queues 277
- LUW (logical unit of work), overview 239

## M

- Macintosh client 78, 81
- main memory buffer 221
- main temporary storage, performance costs 237



- managing regions, destroying a region 19
- managing resources
  - adding a region 273
  - cicsdefault command 272
  - exporting a region 273
  - importing a region 273
  - installing permanent definitions 263
  - Region Definitions (RD) for SFS 59
  - setting up a queue 23, 59, 133
  - setting up a region 17
  - setting up file management 23, 59, 133
  - setting up Program Definitions 127
  - setting up Transaction Definitions 127
- managing resources using cicscp
  - destroying a region 19
  - destroying an SFS server 31
- managing resources using cicssfsdestroy,
  - destroying an SFS server 283
- managing SFS
  - destroying an SFS server 31, 283
- mapsets
  - setup 127, 129
- MaxServer 237
- MD 220
- media archiving
  - disabling 245
  - enabling 245
- Media Recovery Archiving (MRA)
  - advantages of 243
- messages
  - ERZ0070xxE is logged 201
  - ERZ021004E is logged 201
  - ERZ0220xxE is logged 201
  - ERZ0230xxE is logged 201
  - ERZ0250xxE is logged 201
  - ERZ045001I 104
  - ERZ045013E 110
  - ERZ045014E 110
  - ERZ061013E is logged 201
  - ERZ087001E 110
  - NLS (client) 15
  - NLS (region) 21
- mft model 85
- mft-mb model 85
- Micro Focus Server Express COBOL
  - cicsprCOBOL file 175
  - language method file 175
- migrating file and queue management
  - DB2 and SFS 49
  - Oracle and SFS 53
- MinServer 237
- mirrored volumes, hardware fault tolerance 254
- models, autoinstall 85
- Monitoring Data Formatter (cicsmfmt) 221
- Monitoring Definitions (MD), setup 220
- monitoring facility
  - cicsmfmt 221
  - clocks 219
  - contribution of data by application programs 219
  - data-collection control 219
  - event-monitoring points (EMPs) 219

- monitoring facility (*continued*)
  - performance 219
  - performance class 219
  - setup 220
  - user 219
- monitoring, performance costs 236

## N

- National Language Support
  - BMS 128
  - for the client 15
  - for the region 21
  - messages 15, 21
- NetName attribute
  - CTG.INI 81
- netstat command 224
- NLS
  - BMS 128
  - for the client 15
  - for the region 21
  - messages 15, 21
- NLSPATH environment variable
  - setting 12
  - use of 15, 21
- non-XA enabled
  - sample programs 143
- nonrecoverable temporary storage queues 277
- nonrecoverable transient data queues 277
- normal (controlled) shutdown 207

## O

- offline backups 255
- Offline utilities, cicsmfmt 221
- OpThreadPoolSize, improving performance 228
- Oracle
  - backup and recovery 249
  - cicsoraconf command 64
  - configuration procedures 40, 41
  - configuring for queue management 64
  - creating for CICS regions 45
  - enabling the XA interface 156
  - field name restrictions 54
  - file management 40, 41
  - queue and file management 47
  - queue management 40, 41
  - recovery 249
  - requirements 41
  - runtime error 47
  - security considerations 49
  - transaction support, C 156
  - transaction support, COBOL 156
- Oracle (CICS)
  - configuring 44
- OS/2 client 78, 81
- overtyping
  - CEMT screen 122

## P

- password verification
  - using CICS 89
- password, changing 15
- PATH environment variable, setting 12
- PD attributes, use of, RSLKey 103
- performance
  - determining performance
    - constraints 225
    - response time 229
  - reviewing results of tuning 224
  - storage 227
  - symptoms of poor performance 225
  - symptoms of storage stress 226
  - trade-offs 223
  - transaction 219
  - tuning CICS 223
- performance constraints,
  - determining 225
- performance costs 236
- performance information, monitoring facility 219
- performance monitoring
  - information 219
  - setup 220
- performance monitoring user program
  - overview 190
- permanent database
  - sharing 125
  - updating Region Definitions (RD) 115
- PF key values
  - CEMT 121
- physical memory usage, tuning for performance 228
- physically recoverable transient data queues 277
- placement of, cicsprCOBOL 175
- Port attribute (CTG.INI), use of 81
- post-initialization programs
  - considerations when writing 191
  - description 190
- PPC Gateway server
  - availability problems 201
  - startup failures 201
- printer management, configuring
  - Terminal Definitions (WD) 83
- printers
  - setup 83
  - terminals 83
- procedures
  - autoinstalling Program Definitions (PD) 129
  - configuring a region 272
  - configuring DB2 138, 140
  - configuring DB2 for queue management 63
  - configuring File Definitions (FD) 133, 135
  - configuring Journal Definitions (JD) 217, 218
  - configuring Listener Definitions (LD) 82
  - configuring Oracle for queue management 64
  - configuring Program Definitions (PD) 127

procedures (*continued*)

- configuring Schema File Definitions (SCD) 136
- configuring SFS 139, 276
- configuring Temporary Storage Definitions (TSD) 67
- configuring Terminal Definitions (WD) 83, 87
- configuring Transaction Definitions (TD) 130
- configuring Transient Data Definitions (TDD) 68, 69
- configuring User Definitions (UD) 14
- creating SFS logical volumes 274
- creating SFS user ID 273
- SFS
  - configuring 137
  - defining 275
  - starting 276
- Product Definitions (XAD), loading 115
- Program Definitions (PD)
  - adding 127
  - autoinstalling 129
  - changing 127
  - cicsadd 127
  - cicsdelete 127
  - cicsupdate 127
  - configuring 127
  - deleting 127
  - sharing resource definitions 126
- Program Definitions (PD) attributes, use of, RSLKey 103
- program management, autoinstalling
  - Program Definitions (PD) 129
- program management, configuring
  - Program Definitions (PD) 127
- ProgramRSLCheck attribute (RD), use of 101, 107
- programs
  - security 103
  - setup 127, 129
- Protocol attribute
  - CTG.INI 81
  - LD 81
- Protocol attribute (LD), use of 83

## Q

queue management
 

- configuration procedures
  - DB2 31, 32
  - Oracle 40, 41
  - SFS 24
  - SQL 31
- naming of (SQL) 62, 64
- requirements (DB2) 32
- requirements (Oracle) 41
- requirements (SFS) 24

queues
 

- managing resources 23, 59, 133

quiesce stages of normal shutdown 207

## R

RD
 

- change 20

RD (*continued*)
 

- for SFS 59
- setup 20
- SFS servers, sharing resource definitions 125
- sharing resource definitions 125
- shutdown program list 208

RD attributes, use of
 

- EAMLoad 91
- ESMLoad 94, 101
- FileRSLCheck 101, 107
- JournalRSLCheck 101, 107
- LoadDataNumBuckets 227
- ProgramRSLCheck 101, 107
- StartType 280
- TaskSHNumBuckets 227
- TemporaryStorageRSLCheck 107
- TransactionRSLCheck 101, 107
- TransientDataRSLCheck 101, 107
- TSLCheck 94

RD for SFS 59

recoverable resources
 

- backout overview 240
- performance costs 237

recoverable temporary storage
 

- queues 277

recovery
 

- DB2 240, 248
- example 246, 248
- functionality 240
- of autoinstalled terminal definitions 240
- of auxiliary temporary storage queues 240
- of CICS subsystem entries 251
- of CICS user IDs 251
- of configuration 249
- of DB2 247
- of EXEC CICS START PROTECT requests 240
- of intrapartition transient data 240
- of Oracle 249
- of subsystem entries 251
- of user IDs 251
- Oracle 249
- overview 240
- SFS 240, 242, 246

recovery and restart
 

- overview 239
- SFS logical volumes 244
- startup 200

region
 

- cicsdefault command 272
- configuring 272
- default user ID 272
- defining and managing overview 9
- performance cost of multiple 237
- shutdown 207, 210, 211, 212
- starting 280
- startup 199

Region Definitions (RD)
 

- change 20
- for SFS 59
- overriding at startup 280
- setup 20
- SFS servers, sharing resource definitions 125

Region Definitions (RD) (*continued*)
 

- sharing resource definitions 125
- shutdown program list 208

Region Definitions (RD) attributes, use of
 

- EAMLoad 91
- ESMLoad 94, 101
- FileRSLCheck 101, 107
- JournalRSLCheck 101, 107
- LoadDataNumBuckets 227
- ProgramRSLCheck 101, 107
- StartType 280
- TaskSHNumBuckets 227
- TemporaryStorageRSLCheck 107
- TransactionRSLCheck 101, 107
- TransientDataRSLCheck 101, 107
- TSLCheck 94

region storage pool, performance 226, 227

relational database managers
 

- DB2 and the XA interface 37, 150
- enabling the XA interface 150, 156
- Informix and the XA interface 153
- Oracle and the XA interface 47, 156
- queue and file management 37, 47
- setting up the XA interface 153

relational databases, integrating XA
 

- enabled 150, 156

RemoteName attribute, sharing resource definitions 126

requested statistics 214

resource
 

- managing from applications 123
- using CEMT to manage 119

resource definitions
 

- region 17

resource security 106
 

- defining 102
- transaction 102

resource security level (RSL), defined 102

resources
 

- groups of 115
- installing permanent definitions 263
- list of 5
- performance cost of recoverable 237
- security 103
- sharing 125

response time, summary of symptoms and solutions 229

restart file, use of in normal shutdown 208

restart, performance costs 230

RESTORE DATABASE, use of 248, 249

restrictions, field names (DB2) 50

restrictions, field names (Oracle) 54

RPC listener process 233

RS6000
 

- HACMP 255

RSLCheck attribute (TD), use of 101, 102, 107

RSLKey attribute, use of 103

runtime
 

- updating 116

runtime database
 

- description 115
- sharing 125

runtime error, DB2 37

runtime error, Oracle 47

## S

sar command 224

schedtune 229

Schema File Definitions (SCD)

configuring 136

on DB2 138

on SFS 137

security 89

access to CICS-supplied

transactions 96

ATI 95

authentication using CICS 89

authorizing access to transactions 94

CEBR 103

CECI 103

CEDF 103

CEDF debugging transaction 97

CICS authentication 89

CICS-supplied transactions 103

CPMI 103

CSM1 103

CSMI 103

external 188

files 103

journals 103

password verification using CICS 89

programs 103

recovering after a failure 251

recovery of configuration 249

recovery of DB2 247

recovery of Oracle 249

recovery of subsystem entries 251

recovery of user IDs 251

resource 102, 103, 106

resource definition using CICS

commands 99

resources from applications 100

resources outside the runtime 100

service for development and

testing 253

service that owns data 253

service that owns no data 253

startup and shutdown programs 99

supplied transactions 103

temporary storage queues 103

transaction 98

transactions 96, 102, 103

transient data queues 103

user 89

user access to CICS 89, 99

using security keys 102

XA-enabled relational databases 90

Security

hardware faults 253

security considerations, DB2 40

security considerations, Oracle 49

security file manager

availability problems 201

startup failures 201

sequence, startup 197

ServerIdleLimit 237

setup

data files 133

files 133

setup (*continued*)

journals 217

mapsets 127, 129

Monitoring Definitions (MD) 220

monitoring facilities 220

performance monitoring 220

printers 83

programs 127, 129

Region Definitions (RD) 20

TDQs 68

temporary storage queues 66

transactions 130

transient data queues 68

TSQ 66

SFS

availability problems 201

backup 242

cicssdt command 139

cicssfs command 276

cicssfsconf command 276

cicssfscreate command 275

cicssfsimport command 137

configuration procedures 24

configuring 137, 139, 276

configuring TSQ 66

creating logical volumes 274

creating user ID 273

defining 275

file management 24

queue management 24

recovery 240, 242

requirements 24

Schema File Definitions (SCD) 137

shutting down 211, 282

starting 276

starting up 199

startup failures 201

TSQ configuration 66

SFS Server, creating logical volumes

for 274

sharing resource definitions 125

SHLIB\_PATH

environment variables HP-UX 12

SHLIB\_PATH (HP-UX) environment

variable

use of 111

shorthand

resource definitions 8

used for specifying server file

names 125

shutdown

cancel (forced abnormal) 209

CICS region 210

cicsstop (HP or Solaris) 281

forced abnormal 209

immediate 208

normal 207

region 210

software-initiated 209

stopsrc (AIX) 281

uncontrolled 209

shutdown programs, considerations when

writing 192

shutdown statistics 209

shutting down the region 207, 210

shutting down, performance costs 230

signature strings 227

single-phase commit optimization 145

software-initiated shutdown 209

SOSI attribute

specify for DBCS or MBCS

environment 86

SOSI attribute, use of 85

SOSI switch key

specify for DBCS or MBCS

environment 86

SPARC (Solaris), use in backup and

recovery 254

SQL

configuration procedures 31

embedding calls 143

file management 31, 62, 64

naming of 62, 64

queue management 31, 62, 64

starting a region

CICS control program 199

cicscp command 199

cicsstart command 280

procedure 280

startsrc (AIX) command 280

StartType attribute (RD), use of 280

startup

CICS region 199

cicsstart (HP or Solaris)

command 280

invoking user-written programs 191

post-initialization programs 190

recovery and restart 200

region 199

sequence 197

startsrc (AIX) command 280

startup failures, file managers 201

statistics

end-of-day 214

generation 213

interval 214, 216

requested 214

retrieving 214

shutdown 209

unsolicited 214

stopsrc (AIX) command 281

storage

performance 226, 227, 228

violations 227

storage stress condition 226

Structured File Server (SFS)

backing up logical volumes 242

backup 243

copying logical volumes 245

enlarging logical volumes 30

media archiving 245

MRA backup 243

subsystem entries

backup and recovery 251

recovery 251

supplied programs

External Security Manager

(ESM) 188

function shipping 188

IBM CICS Client product

autoinstall 188

performance monitoring user

program 188

terminal autoinstall 188

- supplied programs (*continued*)
  - transaction routing 188
  - user conversion programs 188
- supplied transactions, security 103
- SwitchLoadFile attribute
  - with DB2 63
  - with Oracle 64
- Sybase
  - configure for C transaction support 163
- symbolic links 125
- symptoms, performance 225
- Symrecs file 210
- syncpoint user exit 186
- syncpoints
  - general description 239
  - performance costs 237
- system clock 238

## T

- table space, DB2 39
- task-private storage pool,
  - performance 226, 227
- task-shared storage pool,
  - performance 226, 227
- TCPAddress attribute (LD), use of 83
- TCPProcessCount attribute (LD), use of 83
- TCPService attribute (LD), use of 83
- TD
  - adding 130
  - changing 130
  - cicsadd 130
  - cicsdelete 130
  - cicsupdate 130
  - configuring 130
  - deleting 130
  - start at shutdown 207
- TD attributes, use of
  - RSLCheck 101, 102, 107
  - RSLKey 103
  - TSLKey 95
- TDD attributes, use of
  - RSLKey 103
- TDQ
  - configuring CICS 68
  - configuring Transient Data Definitions (TDD) 68
  - intrapartition 68
  - setup 68
- Telnet client interface
  - methods for setting up 72
  - starting as a foreground process 76
- Telnet server
  - cicscp 71, 73
  - configuring 71
  - destroying 71
  - inetd 71
  - security 76
  - starting 71
  - using 71
- Temporary Storage Definitions (TSD)
  - adding 66
  - changing 66
  - cicsadd 66
  - cicsdelete 66
- Temporary Storage Definitions (TSD) (*continued*)
  - cicsupdate 66
  - configuring 66, 67
  - deleting 66
  - sharing resource definitions 126
- Temporary Storage Definitions (TSD) attributes, use of, RSLKey 103
- temporary storage queues
  - security 103
  - setup 66
- TemporaryStorageRSLCheck attribute (RD), use of 107
- terminal
  - management, configuring Terminal Definitions (WD) 83
  - setup 83
- terminal autoinstall
  - description 188
  - user program 189
- Terminal Definitions (WD)
  - adding 83
  - changing 83
  - cicsadd 83
  - cicsdelete 83
  - cicsupdate 83
  - configuring 83, 87
  - deleting 83
- terminal I/O errors, recovery
  - cancel (forced abnormal) 209
  - terminal error program forced abnormal shutdown 209
  - terminal error program immediate shutdown 208
- terminal type
  - for DBCS or MBCS environment 86
  - specify for DBCS or MBCS environment 86
- termination
  - cancel (forced abnormal) 209
  - forced abnormal 209
  - immediate 208
  - normal (controlled) shutdown 207
  - software-initiated 209
  - uncontrolled 209
- testing system, backup and recovery
  - needed 253
- time zone environment variable 238
- time zones 238
- tools for monitoring CICS, auxiliary trace 221
- TP\_MON\_NAME, DB2 40
- trace
  - auxiliary 221
  - auxiliary storage 221
  - external storage 221
  - main storage 221
  - performance costs 236
- trace table 221
- trace utility program (cicstfmt) 221
- Transaction Definitions (TD)
  - adding 130
  - changing 130
  - cicsadd 130
  - cicsdelete 130
  - cicsupdate 130
  - configuring 130

- Transaction Definitions (TD) (*continued*)
  - deleting 130
  - sharing resource definitions. 126
  - start at shutdown 207
- Transaction Definitions (TD) attributes, use of
  - RSLCheck 101, 102, 107
  - RSLKey 103
- transaction security 98
- transaction security level (TSL), defined 102
- transaction support
  - C and DB2 150
  - C and Oracle 156
  - DB2 and C 150
  - Informix 153
  - Oracle and C 156
  - WebSphere MQ 180
- TransactionRSLCheck attribute (RD), use of 101, 107
- transactions
  - management, configuring Transaction Definitions (TD) 130
  - performance 219
  - resource security 102
  - security 96, 102, 103
  - setup 130
- Transient Data Definitions (TDD)
  - adding 68
  - changing 68
  - cicsadd 68
  - cicsdelete 68
  - cicsupdate 68
  - configuring 68, 69
  - deleting 68
  - sharing resource definitions 126
- Transient Data Definitions (TDD) attributes, use of
  - RSLKey 103
- transient data queues
  - configuring CICS 68
  - configuring Transient Data Definitions (TDD) 68
  - intrapartition 68
  - security 103
  - setup 68
- TransientDataRSLCheck attribute (RD), use of 101, 107
- TSD attributes, use of, RSLKey 103
- TSLCheck attribute (RD), use of 94
- TSLKey attribute (TD), use of 95
- TSLKeyList attribute (UD), use of 95
- TSQ
  - configuring for 66
  - setup 66
- tuning
  - reviewing results of 224
- two-phase commit 132
- types of service, backup and recovery
  - needed 252

## U

- UD
  - adding 14
  - changing 14
  - cicsadd 14

- UD (*continued*)
  - cicsdelete 14
  - cicsupdate 14
  - configuring 14
  - deleting 14
- UD attributes, use of
  - CICSPassword 15
  - RSLKeyList 103
  - TSLKeyList 95
- uncontrolled shutdown 209
- unexpired starts
  - nonprotected 277
  - protected 277
- units of recovery, overview 239
- unsolicited statistics 214
- updating, runtime 116
- UpperCaseSecurity attribute (CTG.INI), use of 81
- user access to cics objects, security 102
- user access to CICS transactions, security 94, 96, 97
- user access to CICS, security 89, 99, 100
- user conversion programs
  - function shipping 188
  - transaction routing 188
- User Definitions (UD)
  - adding 14
  - changing 14
  - cicsadd 14
  - cicsdelete 14
  - cicsupdate 14
  - configuring 14
  - deleting 14
- User Definitions (UD) attributes, use of
  - CICSPassword 15
  - RSLKeyList 103
  - TSLKeyList 95
- user exit
  - dump request 185
  - dynamic distributed program
    - link 185
  - dynamic transaction routing 185
  - implementing 186
  - list of 185
  - migration and coexistence 188
  - naming convention 188
  - overview 185
  - rules for writing programs 187
  - syncpoint 186
  - task termination 185
  - User exit number attribute 186
  - writing programs 187
- user exit number attribute 186
- User exit number attribute 186
- user ID (SFS)
  - creating 273
- user IDs
  - backup and recovery 251
- user management, configuring User Definitions (UD) 14
- user monitoring 219
- user programs
  - External Security Manager (ESM) 188, 189
  - function shipping 188
  - IBM CICS Client product
    - autoinstall 188
- user programs (*continued*)
  - overview 188
  - performance monitoring 188
  - terminal autoinstall 188, 189
  - transaction routing 188
  - user conversion programs 188
- user security 89
- using an External Authentication Manager
  - defining an EAM at startup 94
  - defining the interface 91
  - overview 91
- using an External Security Manager
  - defining an ESM at startup 111
  - defining CICS to an External Security Manager 111
  - defining the interface 107
  - overview 106
- using security keys 102

**V**

- vmstat command 224
- vt100 model 85

**W**

- WebSphere MQ 179
  - building applications 179
  - enabling the XA interface 180
  - environment variables 179
  - installation 179
  - transaction support 180
  - xa support 179
- Windows client 78, 81
- writing user exit programs 187

**X**

- X/Open DTP implementation, security considerations 90
- X/Open XA Interface 143
- XA enabled relational databases
  - enabling the XA interface 150, 156
  - integrating with CICS 150, 156
- XA interface
  - DB2 37, 150
  - enabling for DB2 37, 150
  - enabling for Oracle 47, 156
  - enabling for WebSphere MQ 180
  - Informix 153
  - Oracle 47, 156
  - setting up for Informix 153
  - WebSphere MQ 180
- xa support
  - WebSphere MQ 179
- xa\_open call, use of 90
- XA-enabled relational databases
  - security 90
- XAD, loading 115
- xterm model 85









SC34-6637-00



Spine information:



TXSeries for Multiplatforms

CICS Administration Guide for Open Systems

Version 6.0

SC34-6637-00